A Formal Approach to Physics-Based Attacks in Cyber-Physical Systems*

Ruggero Lanotte[1], Massimo Merro[2], Andrei Munteanu[2], and Luca Viganò[3]

[1]DISUIT, Università dell'Insubria, Italy, ruggero.lanotte@uninsubria.it
[2]Dipartimento di Informatica, Università degli Studi di Verona, Italy, {massimo.merro,andrei.munteanu}@univr.it
[3]Department of Informatics, King's College London, UK, luca.vigano@kcl.ac.uk

Abstract

We apply formal methods to lay and streamline theoretical foundations to reason about Cyber-Physical Systems (CPSs) and physics-based attacks, i.e., attacks targeting physical devices. We focus on a formal treatment of both integrity and denial of service attacks to sensors and actuators of CPSs, and on the timing aspects of these attacks. Our contributions are fourfold. (1) We define a hybrid process calculus to model both CPSs and physics-based attacks. (2) We formalise a threat model that specifies MITM attacks that can manipulate sensor readings or control commands in order to drive a CPS into an undesired state; we group these attacks into classes, and provide the means to assess attack tolerance/vulnerability with respect to a given class of attacks, based on a proper notion of most powerful physics-based attack. (3) We formalise how to estimate the impact of a successful attack on a CPS and investigate possible quantifications of the success chances of an attack. (4) We illustrate our definitions and results by formalising a non-trivial running example in Uppaal SMC, the statistical extension of the Uppaal model checker; we use Uppaal SMC as an automatic tool for carrying out a static security analysis of our running example in isolation and when exposed to three different physics-based attacks with different impacts.

1 Introduction

1.1 Context and motivation

Cyber-Physical Systems (CPSs) are integrations of networking and distributed computing systems with physical processes that monitor and control entities in a physical environment, with feedback loops where physical processes affect computations and vice versa. For example, in real-time control systems, a hierarchy of *sensors*, *actuators* and *control components* are connected to control stations.

In recent years there has been a dramatic increase in the number of attacks to the security of CPSs, e.g., manipulating sensor readings and, in general, influencing physical processes to bring the system into a state desired by the attacker. Some notorious examples are: (i) the *STUXnet* worm, which reprogrammed PLCs of nuclear centrifuges in Iran [35]; (ii) the attack on a sewage treatment facility in Queensland, Australia, which manipulated the SCADA system to release raw sewage into local rivers and parks [53]; (iii) the *BlackEnergy* cyber-attack on the Ukrainian power grid, again compromising the SCADA system [31].

A common aspect of these attacks is that they all compromised *safety critical systems*, i.e., systems whose failures may cause catastrophic consequences. Thus, as stated in [24, 25], the concern for consequences at the

*To appear in ACM Transactions on Privacy and Security (TOPS). Accepted November 22, 2019.

physical level puts *CPS security* apart from standard *information security*, and demands for ad hoc solutions to properly address such novel research challenges.

These ad hoc solutions must explicitly take into consideration a number of specific issues of attacks tailored for CPSs. One main critical issue is the *timing of the attack*: the physical state of a system changes continuously over time and, as the system evolves, some states might be more vulnerable to attacks than others [33]. For example, an attack launched when the target state variable reaches a local maximum (or minimum) may have a great impact on the whole system behaviour, whereas the system might be able to tolerate the same attack if launched when that variable is far from its local maximum or minimum [34]. Furthermore, not only the timing of the attack but also the *duration of the attack* is an important parameter to be taken into consideration in order to achieve a successful attack. For example, it may take minutes for a chemical reactor to rupture [56], hours to heat a tank of water or burn out a motor, and days to destroy centrifuges [35].

Much progress has been done in the last years in developing formal approaches to aid the *safety verification* of CPSs (e.g., [28, 18, 19, 49, 48, 4], to name a few). However, there is still a relatively small number of works that use formal methods in the context of the *security analysis* of CPSs (e.g., [11, 10, 60, 50, 46, 1, 8, 58]). In this respect, to the best of our knowledge, a systematic formal approach to study *physics-based attacks*, that is, attacks targeting the physical devices (sensors and actuators) of CPSs, is still to be fully developed. Our paper moves in this direction by relying on a process calculus approach.

1.2 Background

The dynamic behaviour of the *physical plant* of a CPS is often represented by means of a *discrete-time state-space model*[1] consisting of two equations of the form

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + e_k \end{aligned}$$

where $x_k \in \mathbb{R}^n$ is the current *(physical) state*, $u_k \in \mathbb{R}^m$ is the *input* (i.e., the control actions implemented through actuators) and $y_k \in \mathbb{R}^p$ is the *output* (i.e., the measurements from the sensors). The *uncertainty* $w_k \in \mathbb{R}^n$ and the *measurement error* $e_k \in \mathbb{R}^p$ represent perturbation and sensor noise, respectively, and A, B, and C are matrices modelling the dynamics of the physical system. Here, the *next state* x_{k+1} depends on the current state x_k and the corresponding control actions u_k, at the sampling instant $k \in \mathbb{N}$. The state x_k cannot be directly observed: only its measurements y_k can be observed.

The physical plant is supported by a communication network through which the sensor measurements and actuator data are exchanged with controller(s) and supervisor(s) (e.g., IDSs), which are the *cyber* components (also called *logics*) of a CPS.

1.3 Contributions

In this paper, we focus on a formal treatment of both *integrity* and *Denial of Service (DoS)* attacks to *physical devices* (sensors and actuators) of CPSs, paying particular attention to the *timing aspects* of these attacks. The overall goal of the paper is to apply *formal methodologies* to lay *theoretical foundations* to reason about and formally detect attacks to physical devices of CPSs. A straightforward utilisation of these methodologies is for *model-checking* (as, e.g., in [19]) or *monitoring* (as, e.g., in [4]) in order to be able to verify security properties of CPSs either before system deployment or, when static analysis is not feasible, at runtime to promptly detect undesired behaviours. In other words, we aim at providing an essential stepping stone for formal and automated analysis techniques for checking the security of CPSs (rather than for providing defence techniques, i.e., *mitigation* [45]).

Our contribution is fourfold. The *first contribution* is the definition of a *hybrid process calculus*, called `CCPSA`, to formally specify both CPSs and *physics-based attacks*. In `CCPSA`, CPSs have two components:

[1] See [62, 63] for a taxonomy of the time-scale models used to represent CPSs.

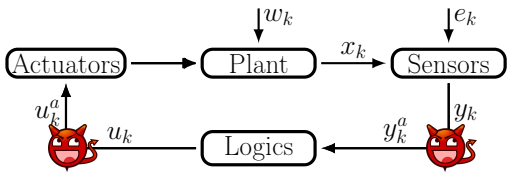

Figure 1: MITM attacks to sensor readings and control commands

- a *physical component* denoting the *physical plant* (also called environment) of the system, and containing information on state variables, actuators, sensors, evolution law, etc., and
- a *cyber component* that governs access to sensors and actuators, and channel-based communication with other cyber components.

Thus, channels are used for logical interactions between cyber components, whereas sensors and actuators make possible the interaction between cyber and physical components.

`CCPSA` adopts a *discrete notion of time* [27] and it is equipped with a *labelled transition semantics (LTS)* that allows us to observe both *physical events* (system deadlock and violations of safety conditions) and *cyber events* (channel communications). Based on our LTS, we define two *compositional* trace-based system preorders: a deadlock-sensitive *trace preorder*, $\sqsubseteq$, and a *timed variant*, $\sqsubseteq_{m..n}$, for $m \in \mathbb{N}^+$ and $n \in \mathbb{N}^+ \cup \{\infty\}$, which takes into account discrepancies of execution traces within the discrete time interval $m..n$. Intuitively, given two CPSs Sys_1 and Sys_2, we write $Sys_1 \sqsubseteq_{m..n} Sys_2$ if Sys_2 simulates the execution traces of Sys_1, except for the time interval $m..n$; if $n = \infty$ then the simulation only holds for the first $m-1$ time slots.

As a *second contribution*, we formalise a *threat model* that specifies *man-in-the-middle (MITM) attacks* that can manipulate *sensor readings* or *control commands* in order to drive a CPS into an undesired state [55].[2] Without loss of generality, MITM attacks targeting physical devices (sensors or actuators) can be assimilated to *physical attacks*, i.e., those attacks that directly compromise physical devices (e.g., electromagnetic attacks). As depicted in Figure 1, our attacks may affect directly the sensor measurements or the controller commands:

- *Attacks on sensors* consist of reading and eventually replacing y_k (the sensor measurements) with y_k^a.
- *Attacks on actuators* consist of reading, dropping and eventually replacing the controller commands u_k with u_k^a, affecting directly the actions the actuators may execute.

We group attacks into classes. A *class of attacks* takes into account both the potential malicious activities $\mathcal{I}$ on physical devices and the *timing parameters* m and n of the attack: begin and end of the attack. We represent a class C as a total function $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$. Intuitively, for $\iota \in \mathcal{I}$, $C(\iota) \subseteq m..n$ denotes the set of time instants when an attack of class C may tamper with the device ι.

In order to make security assessments on our CPSs, we adopt a well-known approach called *Generalized Non Deducibility on Composition (GNDC)* [17]. Thus, in our calculus `CCPSA`, we say that a CPS *Sys tolerates* an attack A if

$$Sys \parallel A \sqsubseteq Sys\,.$$

In this case, the presence of the attack A, does not change the (physical and logical) observable behaviour of the system *Sys*, and the attack can be considered harmless.

On the other hand, we say that a CPS *Sys* is *vulnerable* to an attack A of class $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$ if there is a time interval $m'..n'$ in which the attack becomes observable (obviously, $m' \geq m$). Formally, we write:

$$Sys \parallel A \sqsubseteq_{m'..n'} Sys\,.$$

We provide sufficient criteria to prove attack tolerance/vulnerability to attacks of an arbitrary class C. We define a notion of *most powerful physics-based attack* of a given class C, $Top(C)$, and prove that if a CPS

[2]Note that we focus on attackers who have already entered the CPS, and we do not consider how they gained access to the system, e.g., by attacking an Internet-accessible controller or one of the communication protocols as a Dolev-Yao-style attacker [16] would do.

tolerates $Top(C)$ then it tolerates all attacks A of class C (and of any weaker class[3]). Similarly, if a CPS is vulnerable to $Top(C)$, in the time interval $m'..n'$, then no attacks of class C (or weaker) can affect the system out of that time interval. This is very useful when checking for attack tolerance/vulnerability with respect to all attacks of a given class C.

As a *third contribution*, we formalise how to estimate the *impact of a successful attack* on a CPS. As expected, *risk assessment* in industrial CPSs is a crucial phase preceding any defence strategy implementation [54]. The objective of this phase is to prioritise among vulnerabilities; this is done based on the likelihood that vulnerabilities are exploited, and the impact on the system under attack if exploitation occurs. In this manner, the resources can then be focused on preventing the most critical vulnerabilities [44]. We provide a *metric* to estimate the *maximum perturbation* introduced in the system under attack with respect to its genuine behaviour, according to its evolution law and the uncertainty of the model. Then, we prove that the impact of the most powerful attack $Top(C)$ represents an upper bound for the impact of any attack A of class C (or weaker).

Finally, as a *fourth contribution*, we formalise a *running example* in Uppaal SMC [15], the statistical extension of the Uppaal model checker [5] supporting the analysis of systems expressed as composition of *timed and/or probabilistic automata*. Our goal is to test Uppaal SMC as an automatic tool for the *static security analysis* of a simple but significant CPS exposed to a number of different physics-based attacks with different impacts on the system under attack. Here, we wish to remark that while we have kept our running example simple, it is actually non-trivial and designed to describe a wide number of attacks, as will become clear below.

This paper extends and supersedes a preliminary conference version that appeared in [40]. All the results presented in the current paper have been formally proven, although, due to lack of space, proofs of minor statements can be found in the associated technical report [39]. The Uppaal SMC models of our system and the attacks that we have found are available at the repository `https://bitbucket.org/AndreiMunteanu/cps_smc/src/`.

1.4 Organisation

In Section 2, we give syntax and semantics of `CCPSA`. In Section 3, we provide our running example and its formalisation in Uppaal SMC. In Section 4, we first define our threat model for physics-based attacks, then we use Uppaal SMC to carry out a security analysis of our running example when exposed to three different attacks, and, finally, we provide sufficient criteria for attack tolerance/vulnerability, based on a proper notion of most powerful attack. In Section 5, we estimate the impact of attacks on CPSs and prove that the most powerful attack of a given class has the maximum impact with respect to all attacks of the same class (or of a weaker one). In Section 6, we draw conclusions and discuss related and future work.

2 The Calculus

In this section, we introduce our *Calculus of Cyber-Physical Systems and Attacks*, `CCPSA`, which extends the *Calculus of Cyber-Physical Systems*, defined in our companion papers [37, 42], with specific features to formalise and study attacks to physical devices.

Let us start with some preliminary notation.

2.1 Syntax of `CCPSA`

Notation 1. *We use x, x_k for* state variables *(associated to physical states of systems), c, d for* communication channels*, a, a_k for* actuator devices*, and s, s_k or* sensors devices*.*

Actuator names *are metavariables for actuator devices like valve, light, etc. Similarly,* sensor names *are metavariables for sensor devices, e.g., a sensor thermometer that measures a state variable called temperature, with a given precision.*

[3]Intuitively, attacks of classes weaker than C can do less with respect to attacks of class C.

Values, *ranged over by* v, v', w, *are built from basic values, such as Booleans, integers and real numbers; they also include names.*

Given a generic set of names $\mathcal{N}$*, we write* $\mathbb{R}^{\mathcal{N}}$ *to denote the set of functions assigning a real value to each name in* $\mathcal{N}$*. For* $\xi \in \mathbb{R}^{\mathcal{N}}$*,* $n \in \mathcal{N}$ *and* $v \in \mathbb{R}$*, we write* $\xi[n \mapsto v]$ *to denote the function* $\psi \in \mathbb{R}^{\mathcal{N}}$ *such that* $\psi(m) = \xi(m)$*, for any* $m \neq n$*, and* $\psi(n) = v$*. Given two generic functions* ξ_1 *and* ξ_2 *with disjoint domains* $\mathcal{N}_1$ *and* $\mathcal{N}_2$*, respectively, we denote with* $\xi_1 \cup \xi_2$ *the function such that* $(\xi_1 \cup \xi_2)(n) = \xi_1(n)$*, if* $n \in \mathcal{N}_1$*, and* $(\xi_1 \cup \xi_2)(n) = \xi_2(n)$*, if* $n \in \mathcal{N}_2$*.*

In general, a cyber-physical system consists of: (i) a *physical component* (defining state variables, physical devices, physical evolution, etc.) and (ii) a *cyber (or logical) component* that interacts with the physical devices (sensors and actuators) and communicates with other cyber components of the same or of other CPSs.

Physical components in `CCPSA` are given by two sub-components: (i) the *physical state*, which is supposed to change at runtime, and (ii) the *physical environment*, which contains static information.[4]

Definition 1 (Physical state)**.** *Let* $\mathcal{X}$ *be a set of state variables,* $\mathcal{S}$ *be a set of sensors, and* $\mathcal{A}$ *be a set of actuators. A* physical state S *is a triple* $\langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle$*, where:*

- $\xi_{\mathrm{x}} \in \mathbb{R}^{\mathcal{X}}$ *is the* state function*,*
- $\xi_{\mathrm{s}} \in \mathbb{R}^{\mathcal{S}}$ *is the* sensor function*,*
- $\xi_{\mathrm{a}} \in \mathbb{R}^{\mathcal{A}}$ *is the* actuator function*.*

All functions defining a physical state are total*.*

The *state function* ξ_{x} returns the current value associated to each variable in $\mathcal{X}$, the *sensor function* ξ_{s} returns the current value associated to each sensor in $\mathcal{S}$ and the *actuator function* ξ_{a} returns the current value associated to each actuator in $\mathcal{A}$.

Definition 2 (Physical environment)**.** *Let* $\mathcal{X}$ *be a set of state variables,* $\mathcal{S}$ *be a set of sensors, and* $\mathcal{A}$ *be a set of actuators. A* physical environment E *is a 6-tuple* $\langle evol, meas, inv, safe, \xi_{\mathrm{w}}, \xi_{\mathrm{e}} \rangle$*, where:*

- $evol : \mathbb{R}^{\mathcal{X}} \times \mathbb{R}^{\mathcal{A}} \times \mathbb{R}^{\mathcal{X}} \to 2^{\mathbb{R}^{\mathcal{X}}}$ *is the* evolution map*,*
- $meas : \mathbb{R}^{\mathcal{X}} \times \mathbb{R}^{\mathcal{S}} \to 2^{\mathbb{R}^{\mathcal{S}}}$ *is the* measurement map*,*
- $inv \in 2^{\mathbb{R}^{\mathcal{X}}}$ *is the* invariant set*,*
- $safe \in 2^{\mathbb{R}^{\mathcal{X}}}$ *is the* safety set*,*
- $\xi_{\mathrm{w}} \in \mathbb{R}^{\mathcal{X}}$ *is the* uncertainty function*,*
- $\xi_{\mathrm{e}} \in \mathbb{R}^{\mathcal{S}}$ *is the* sensor-error function*.*

All functions defining a physical environment are total functions*.*

The *evolution map evol* models the *evolution law* of the physical system, where changes made on actuators may reflect on state variables. Given a state function, an actuator function, and an uncertainty function, the *evolution map evol* returns the set of next *admissible state functions*. Since we assume an *uncertainty* in our models, *evol* does not return a single state function but a set of possible state functions.

The *measurement map meas* returns the set of next *admissible sensor functions* based on the current state function. Since we assume error-prone sensors, *meas* does not return a single sensor function but a set of possible sensor functions.

The *invariant set inv* represents the set of state functions that satisfy the invariant of the system. A CPS that gets into a physical state with a state function that does not satisfy the invariant is in *deadlock*.

[4] Actually, this information is periodically updated (say, every six months) to take into account possible drifts of the system.

Similarly, the *safety set safe* represents the set of state functions that satisfy the safety conditions of the system. Intuitively, if a CPS gets into an unsafe state, then its functionality may get compromised.

The *uncertainty function* ξ_{w} returns the uncertainty (or accuracy) associated to each state variable. Thus, given a state variable $x \in \mathcal{X}$, $\xi_{\mathrm{w}}(x)$ returns the maximum distance between the real value of x, in an arbitrary moment in time, and its representation in the model. For $\xi_{\mathrm{w}}, \xi'_{\mathrm{w}} \in \mathbb{R}^{\mathcal{X}}$, we will write $\xi_{\mathrm{w}} \leq \xi'_{\mathrm{w}}$ if $\xi_{\mathrm{w}}(x) \leq \xi'_{\mathrm{w}}(x)$, for any $x \in \mathcal{X}$. The evolution map *evol* is obviously *monotone* with respect to uncertainty: if $\xi_{\mathrm{w}} \leq \xi'_{\mathrm{w}}$ then $evol(\xi_{\mathrm{x}}, \xi_{\mathrm{a}}, \xi_{\mathrm{w}}) \subseteq evol(\xi_{\mathrm{x}}, \xi_{\mathrm{a}}, \xi'_{\mathrm{w}})$.

Finally, the *sensor-error function* ξ_{e} returns the maximum error associated to each sensor in $\mathcal{S}$.

Let us now define formally the cyber component of a CPS in `CCPSA`. Our (logical) processes build on Hennessy and Regan's *Timed Process Language TPL* [27], basically, CCS enriched with a discrete notion of time. We extend TPL with two main ingredients:

- two constructs to read values detected at sensors and write values on actuators, respectively;
- special constructs to represent malicious activities on physical devices.

The remaining constructs are the same as those of TPL.

Definition 3 (Processes)**.** Processes *are defined as follows:*

$$
\begin{array}{rcl}
P, Q & ::= & \mathsf{nil} \mid \mathsf{tick}.P \mid P \parallel Q \mid \pi.P \mid \phi.P \mid \lfloor \mu.P \rfloor Q \mid \mathsf{if}\,(b)\,\{P\}\,\mathsf{else}\,\{Q\} \mid P\backslash c \mid H\langle \tilde{w} \rangle \\
\pi & ::= & \mathsf{rcv}\, c(x) \mid \mathsf{snd}\, c\langle v \rangle \\
\phi & ::= & \mathsf{read}\, s(x) \mid \mathsf{write}\, a\langle v \rangle \\
\mu & ::= & \mathsf{sniff}\, s(x) \mid \mathsf{drop}\, a(x) \mid \mathsf{forge}\, p\langle v \rangle\,.
\end{array}
$$

We write nil for the *terminated process*. The process $\mathsf{tick}.P$ sleeps for one time unit and then continues as P. We write $P \parallel Q$ to denote the *parallel composition* of concurrent *threads* P and Q. The process $\pi.P$ denotes *channel transmission*. The construct $\phi.P$ denotes activities on *physical devices*, i.e., *sensor reading* and *actuator writing*. The process $\lfloor \mu.P \rfloor Q$ denotes MITM malicious activities under timeout targeting physical devices (sensors and actuators). More precisely, we support *sensor sniffing*, *drop of actuator commands*, and *integrity attacks on data coming from sensors and addressed to actuators*. Thus, for instance, $\lfloor \mathsf{drop}\, a(x).P \rfloor Q$ drops a command on the actuator a supplied by the controller in the current time slot; otherwise, if there are no commands on a, it moves to the next time slot and evolves into Q.

The process $P\backslash c$ is the channel restriction operator of CCS. We sometimes write $P\backslash\{c_1, c_2, \ldots, c_n\}$ to mean $P\backslash c_1\backslash c_2 \cdots \backslash c_n$. The process $\mathsf{if}\,(b)\,\{P\}\,\mathsf{else}\,\{Q\}$ is the standard conditional, where b is a decidable guard. In processes of the form $\mathsf{tick}.Q$ and $\lfloor \mu.P \rfloor Q$, the occurrence of Q is said to be *time-guarded*. The process $H\langle \tilde{w} \rangle$ denotes (guarded) recursion.

We assume a set of *process identifiers* ranged over by H, H_1, H_2. We write $H\langle w_1, \ldots, w_k \rangle$ to denote a recursive process H defined via an equation $H(x_1, \ldots, x_k) = P$, where (i) the tuple $x_1, \ldots, x_k$ contains all the variables that appear free in P, and (ii) P contains only guarded occurrences of the process identifiers, such as H itself. We say that recursion is *time-guarded* if P contains only time-guarded occurrences of the process identifiers. Unless explicitly stated our recursive processes are always time-guarded.

In the constructs $\mathsf{rcv}\, c(x).P$, $\mathsf{read}\, s(x).P$, $\lfloor \mathsf{sniff}\, s(x).P \rfloor Q$ and $\lfloor \mathsf{drop}\, a(x).P \rfloor Q$ the variable x is said to be *bound*. This gives rise to the standard notions of *free/bound (process) variables* and *α-conversion*. A term is *closed* if it does not contain free variables, and we assume to always work with closed processes: the absence of free variables is preserved at run-time. As further notation, we write $T\{^{v}/_{x}\}$ for the substitution of all occurrences of the free variable x in T with the value v.

Everything is in place to provide the definition of cyber-physical systems expressed in `CCPSA`.

Definition 4 (Cyber-physical system)**.** *Fixed a set of state variables $\mathcal{X}$, a set of sensors $\mathcal{S}$, and a set of actuators $\mathcal{A}$, a* cyber-physical system *in `CCPSA` is given by two main components:*

- *a* physical component *consisting of*
  - *a* physical environment *E defined on $\mathcal{X}$, $\mathcal{S}$, and $\mathcal{A}$, and*

– *a* physical state S *recording the current values associated to the state variables in* $\mathcal{X}$*, the sensors in* $\mathcal{S}$*, and the actuators in* $\mathcal{A}$*;*

- *a* cyber component P *that interacts with the sensors in* $\mathcal{S}$ *and the actuators* $\mathcal{A}$*, and can communicate, via channels, with other cyber components of the same or of other CPSs.*

We write $E; S \bowtie P$ *to denote the resulting CPS, and use* M *and* N *to range over CPSs. Sometimes, when the physical environment* E *is clearly identified, we write* $S \bowtie P$ *instead of* $E; S \bowtie P$*. CPSs of the form* $S \bowtie P$ *are called environment-free CPSs.*

The syntax of our CPSs is slightly too permissive as a process might use sensors and/or actuators that are not defined in the physical state. To rule out ill-formed CPSs, we use the following definition.

Definition 5 (Well-formedness)**.** *Let* $E = \langle evol, meas, inv, safe, \xi_{\mathrm{w}}, \xi_{\mathrm{e}} \rangle$ *be a physical environment, let* $S = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle$ *be a physical state defined on a set of physical variables* $\mathcal{X}$*, a set of sensors* $\mathcal{S}$*, and a set of actuators* $\mathcal{A}$*, and let* P *be a process. The CPS* $E; S \bowtie P$ *is said to be* well-formed *if: (i) any sensor mentioned in* P *is in the domain of the function* ξ_{s}*; (ii) any actuator mentioned in* P *is in the domain of the function* ξ_{a}*.*

In the rest of the paper, we will always work with well-formed CPSs and use the following abbreviations.

Notation 2. *We write* $\mu.P$ *for the process defined via the equation* $Q = \lfloor \mu.P \rfloor Q$*, where* Q *does not occur in* P*. Further, we write*

- $\lfloor \mu \rfloor Q$ *as an abbreviation for* $\lfloor \mu.\mathsf{nil} \rfloor Q$*,*
- $\lfloor \mu.P \rfloor$ *as an abbreviation for* $\lfloor \mu.P \rfloor \mathsf{nil}$*,*
- $\mathsf{snd}\, c$ *and* $\mathsf{rcv}\, c$*, when channel* c *is used for pure synchronisation,*
- $\mathsf{tick}^k.P$ *as a shorthand for* $\mathsf{tick} \ldots \mathsf{tick}.P$*, where the prefix* tick *appears* $k \geq 0$ *consecutive times.*

Finally, let $M = E; S \bowtie P$*, we write* $M \parallel Q$ *for* $E; S \bowtie (P \parallel Q)$*, and* $M \backslash c$ *for* $E; S \bowtie (P \backslash c)$*.*

2.2 Labelled transition semantics

In this subsection, we provide the dynamics of `CCPSA` in terms of a *labelled transition system (LTS)* in the SOS style of Plotkin. First, we give in Table 1 an LTS for logical processes, then in Table 2 we lift transition rules from processes to environment-free CPSs.

In Table 1, the meta-variable λ ranges over labels in the set $\{\mathsf{tick}, \tau, \overline{c}v, cv, a!v, s?v, \lightning p!v, \lightning p?v\}$. Rules (Outp), (Inpp) and (Com) serve to model channel communication, on some channel c. Rules (Read) and (Write) denote sensor reading and actuator writing, respectively. The following three rules model three different MITM malicious activities: sensor sniffing, dropping of actuator commands, and integrity attacks on data coming from sensors or addressed to actuators. In particular, rule ($\lightning$ActDrop$\lightning$) models a *DoS attack to the actuator* a, where the update request of the controller is dropped by the attacker and it never reaches the actuator, whereas rule ($\lightning$SensIntegr$\lightning$) models an *integrity attack on sensor* s, as the controller of s is supplied with a fake value v forged by the attack. Rule (Par) propagates untimed actions over parallel components. Rules (Res), (Rec), (Then) and (Else) are standard. The following four rules (TimeNil), (Sleep), (TimeOut) and (TimePar) model the passage of time. For simplicity, we omit the symmetric counterparts of the rules (Com), ($\lightning$ActDrop$\lightning$), ($\lightning$SensIntegr$\lightning$), and (Par).

In Table 2, we lift the transition rules from processes to environment-free CPSs of the form $S \bowtie P$ for $S = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle$. The transition rules are parametric on a physical environment E. Except for rule (Deadlock), all rules have a common premise $\xi_{\mathrm{x}} \in inv$: a system can evolve only if the invariant is satisfied by the current physical state. Here, actions, ranged over by α, are in the set $\{\tau, \overline{c}v, cv, \mathsf{tick}, \mathsf{deadlock}, \mathsf{unsafe}\}$. These actions denote: internal activities (τ); channel transmission ($\overline{c}v$ and cv); the passage of time (tick); and two specific physical events: system deadlock ($\mathsf{deadlock}$) and the violation of the safety conditions (unsafe). Rules (Out) and (Inp) model transmission and reception, with an external system, on a channel c. Rule (SensRead) models the reading of the current data detected at a sensor s; here, the presence of a malicious

Table 1: LTS for processes

$$\text{(Inpp)}\ \frac{-}{\mathsf{rcv}\, c(x).P \xrightarrow{cv} P\{^{v}/_{x}\}} \qquad \text{(Outp)}\ \frac{-}{\mathsf{snd}\, c\langle v\rangle.P \xrightarrow{\overline{c}v} P}$$

$$\text{(Com)}\ \frac{P \xrightarrow{\overline{c}v} P' \quad Q \xrightarrow{cv} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \qquad \text{(Par)}\ \frac{P \xrightarrow{\lambda} P' \quad \lambda \neq \mathsf{tick}}{P \parallel Q \xrightarrow{\lambda} P' \parallel Q}$$

$$\text{(Read)}\ \frac{-}{\mathsf{read}\, s(x).P \xrightarrow{s?v} P\{^{v}/_{x}\}} \qquad \text{(Write)}\ \frac{-}{\mathsf{write}\, a\langle v\rangle.P \xrightarrow{a!v} P}$$

$$\text{(}\lightning\text{Sniff}\lightning\text{)}\ \frac{-}{\lfloor \mathsf{sniff}\, s(x).P \rfloor Q \xrightarrow{\lightning s?v} P\{^{v}/_{x}\}} \qquad \text{(}\lightning\text{Drop}\lightning\text{)}\ \frac{-}{\lfloor \mathsf{drop}\, a(x).P \rfloor Q \xrightarrow{\lightning a?v} P\{^{v}/_{x}\}}$$

$$\text{(}\lightning\text{Forge}\lightning\text{)}\ \frac{p \in \{s, a\}}{\lfloor \mathsf{forge}\, p\langle v\rangle.P \rfloor Q \xrightarrow{\lightning p!v} P}$$

$$\text{(}\lightning\text{ActDrop}\lightning\text{)}\ \frac{P \xrightarrow{a!v} P' \quad Q \xrightarrow{\lightning a?v} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \qquad \text{(}\lightning\text{SensIntegr}\lightning\text{)}\ \frac{P \xrightarrow{\lightning s!v} P' \quad Q \xrightarrow{s?v} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$$

$$\text{(Res)}\ \frac{P \xrightarrow{\lambda} P' \quad \lambda \notin \{cv, \overline{c}v\}}{P\backslash c \xrightarrow{\lambda} P'\backslash c} \qquad \text{(Rec)}\ \frac{P\{^{\tilde{w}}/_{\tilde{x}}\} \xrightarrow{\lambda} Q \quad H(\tilde{x}) = P}{H\langle \tilde{w}\rangle \xrightarrow{\lambda} Q}$$

$$\text{(Then)}\ \frac{[\![b]\!] = \mathsf{true} \quad P \xrightarrow{\lambda} P'}{\mathsf{if}\ (b)\ \{P\}\ \mathsf{else}\ \{Q\} \xrightarrow{\lambda} P'} \qquad \text{(Else)}\ \frac{[\![b]\!] = \mathsf{false} \quad Q \xrightarrow{\lambda} Q'}{\mathsf{if}\ (b)\ \{P\}\ \mathsf{else}\ \{Q\} \xrightarrow{\lambda} Q'}$$

$$\text{(TimeNil)}\ \frac{-}{\mathsf{nil} \xrightarrow{\mathsf{tick}} \mathsf{nil}} \qquad \text{(Sleep)}\ \frac{-}{\mathsf{tick}.P \xrightarrow{\mathsf{tick}} P}$$

$$\text{(Timeout)}\ \frac{-}{\lfloor \mu.P \rfloor Q \xrightarrow{\mathsf{tick}} Q} \qquad \text{(TimePar)}\ \frac{P \xrightarrow{\mathsf{tick}} P' \quad Q \xrightarrow{\mathsf{tick}} Q'}{P \parallel Q \xrightarrow{\mathsf{tick}} P' \parallel Q'}$$

action $\lightning s!w$ would prevent the reading of the sensor. We already said that rule ($\lightning$SensIntegr$\lightning$) of Table 1 models integrity attacks on a sensor s. However, together with rule (SensRead), it also serves to implicitly model *DoS attacks on a sensor s*, as the controller of s cannot read its correct value if the attacker is currently supplying a fake value for it. Rule ($\lightning$SensSniff$\lightning$) allows the attacker to read the confidential value detected at a sensor s. Rule (ActWrite) models the writing of a value v on an actuator a; here, the presence of an attack capable of performing a drop action $\lightning a?v$ prevents the access to the actuator by the controller. Rule ($\lightning$ActIntegr$\lightning$) models a *MITM integrity attack to an actuator a*, as the actuator is provided with a value forged by the attack. Rule (Tau) lifts non-observable actions from processes to systems. This includes communications channels and attacks' accesses to physical devices. A similar lifting occurs in rule (Time) for timed actions, where $\text{next}(E; S)$ returns the set of possible physical states for the next time slot. Formally, for $E = \langle evol, meas, inv, safe, \xi_{\mathrm{w}}, \xi_{\mathrm{e}}\rangle$ and $S = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}}\rangle$, we define:

$$\text{next}(E; S) \stackrel{\text{def}}{=} \left\{ \langle \xi'_{\mathrm{x}}, \xi'_{\mathrm{s}}, \xi'_{\mathrm{a}}\rangle \,:\, \xi'_{\mathrm{x}} \in evol(\xi_{\mathrm{x}}, \xi_{\mathrm{a}}, \xi_{\mathrm{w}}) \,\wedge\, \xi'_{\mathrm{s}} \in meas(\xi'_{\mathrm{x}}, \xi_{\mathrm{e}}) \,\wedge\, \xi'_{\mathrm{a}} = \xi_{\mathrm{a}} \right\}.$$

Thus, by an application of rule (Time) a CPS moves to the next physical state, in the next time slot. Rule (Deadlock) is introduced to signal the violation of the invariant. When the invariant is violated, a system deadlock occurs and then, in CCPSA, the system emits a special action deadlock, forever. Similarly, rule (Safety) is introduced to detect the violation of safety conditions. In this case, the system may emit a special action unsafe and then continue its evolution.

Summarising, in the LTS of Table 2 we define transitions rules of the form $S \bowtie P \xrightarrow{\alpha} S' \bowtie P'$, parametric

Table 2: LTS for CPSs $S \bowtie P$ parametric on an environment $E = \langle evol, meas, inv, safe, \xi_{\mathrm{w}}, \xi_{\mathrm{e}} \rangle$

$$\text{(Out)}\ \frac{S = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \quad P \xrightarrow{\overline{c}v} P' \quad \xi_{\mathrm{x}} \in inv}{S \bowtie P \xrightarrow{\overline{c}v} S \bowtie P'} \qquad \text{(Inp)}\ \frac{S = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \quad P \xrightarrow{cv} P' \quad \xi_{\mathrm{x}} \in inv}{S \bowtie P \xrightarrow{cv} S \bowtie P'}$$

$$\text{(SensRead)}\ \frac{P \xrightarrow{s?v} P' \quad \xi_{\mathrm{s}}(s) = v \quad P \xrightarrow{\lightning s!v} \not\rightarrow \quad \xi_{\mathrm{x}} \in inv}{\langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \bowtie P \xrightarrow{\tau} \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \bowtie P'}$$

$$\text{(}\lightning\text{SensSniff}\lightning\text{)}\ \frac{P \xrightarrow{\lightning s?v} P' \quad \xi_{\mathrm{s}}(s) = v \quad \xi_{\mathrm{x}} \in inv}{\langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \bowtie P \xrightarrow{\tau} \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \bowtie P'}$$

$$\text{(ActWrite)}\ \frac{P \xrightarrow{a!v} P' \quad \xi'_{\mathrm{a}} = \xi_{\mathrm{a}}[a \mapsto v] \quad P \xrightarrow{\lightning a?v} \not\rightarrow \quad \xi_{\mathrm{x}} \in inv}{\langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \bowtie P \xrightarrow{\tau} \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi'_{\mathrm{a}} \rangle \bowtie P'}$$

$$\text{(}\lightning\text{ActIntegr}\lightning\text{)}\ \frac{P \xrightarrow{\lightning a!v} P' \quad \xi'_{\mathrm{a}} = \xi_{\mathrm{a}}[a \mapsto v] \quad \xi_{\mathrm{x}} \in inv}{\langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \bowtie P \xrightarrow{\tau} \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi'_{\mathrm{a}} \rangle \bowtie P'}$$

$$\text{(Tau)}\ \frac{P \xrightarrow{\tau} P' \quad \xi_{\mathrm{x}} \in inv}{\langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \bowtie P \xrightarrow{\tau} \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \bowtie P'} \qquad \text{(Deadlock)}\ \frac{S = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \quad \xi_{\mathrm{x}} \notin inv}{S \bowtie P \xrightarrow{\mathsf{deadlock}} S \bowtie P}$$

$$\text{(Time)}\ \frac{P \xrightarrow{\mathsf{tick}} P' \quad S = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \quad S' \in \mathrm{next}(E; S) \quad \xi_{\mathrm{x}} \in inv}{S \bowtie P \xrightarrow{\mathsf{tick}} S' \bowtie P'}$$

$$\text{(Safety)}\ \frac{S = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle \quad \xi_{\mathrm{x}} \notin safe \quad \xi_{\mathrm{x}} \in inv}{S \bowtie P \xrightarrow{\mathsf{unsafe}} S \bowtie P}$$

on some physical environment E. As physical environments do not change at runtime, $S \bowtie P \xrightarrow{\alpha} S' \bowtie P'$ entails $E; S \bowtie P \xrightarrow{\alpha} E; S' \bowtie P'$, thus providing the LTS for all CPSs in `CCPSA`.

Remark 1. *Note that our operational semantics ensures that malicious actions of the form $\lightning s!v$ (integrity/DoS attack on sensor s) or $\lightning a?v$ (DoS attack on actuator a) have a pre-emptive power. These attacks can always prevent the regular access to a physical device by its controller.*

2.3 Behavioural semantics

Having defined the actions that can be performed by a CPS of the form $E; S \bowtie P$, we can easily concatenate these actions to define the possible *execution traces* of the system. Formally, given a trace $t = \alpha_1 \dots \alpha_n$, we will write $\xrightarrow{t}$ as an abbreviation for $\xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n}$, and we will use the function $\#\mathsf{tick}(t)$ to get the number of occurrences of the action tick in t.

The notion of trace allows us to provide a formal definition of system soundness: a CPS is said to be *sound* if it never deadlocks and never violates the safety conditions.

Definition 6 (System soundness)**.** *Let M be a well-formed CPS. We say that M is* sound *if whenever $M \xrightarrow{t} M'$, for some t, the actions* deadlock *and* unsafe *never occur in t.*

In our security analysis, we will always focus on sound CPSs.

We recall that the *observable activities* in `CCPSA` are: time passing, system deadlock, violation of safety conditions, and channel communication. Having defined a labelled transition semantics, we are ready to formalise our behavioural semantics, based on execution traces.

We adopt a standard notation for weak transitions: we write $\Rightarrow$ for $(\xrightarrow{\tau})^*$, whereas $\overset{\alpha}{\Longrightarrow}$ means $\Longrightarrow \xrightarrow{\alpha} \Longrightarrow$, and finally $\overset{\hat{\alpha}}{\Longrightarrow}$ denotes $\Rightarrow$ if $\alpha = \tau$ and $\overset{\alpha}{\Longrightarrow}$ otherwise. Given a trace $t = \alpha_1 \dots \alpha_n$, we write

$\stackrel{\hat{t}}{\Longrightarrow}$ as an abbreviation for $\stackrel{\widehat{\alpha_1}}{\Longrightarrow} \ldots \stackrel{\widehat{\alpha_n}}{\Longrightarrow}$.

Definition 7 (Trace preorder)**.** *We write $M \sqsubseteq N$ if whenever $M \stackrel{t}{\longrightarrow} M'$, for some t, there is N' such that $N \stackrel{\hat{t}}{\Longrightarrow} N'$.*

Remark 2. *Unlike other process calculi, in* `CCPSA` *our trace preorder is able to observe (physical) deadlock due to the presence of the rule (Deadlock) and the special action* deadlock*: whenever $M \sqsubseteq N$ then M eventually deadlocks if and only if N eventually deadlocks (see Lemma 1 in the appendix).*

Our trace preorder can be used for *compositional reasoning* in those contexts that don't interfere on physical devices (sensors and actuators) while they may interfere on logical components (via channel communication). In particular, trace preorder is preserved by parallel composition of *physically-disjoint* CPSs, by parallel composition of *pure-logical* processes, and by channel restriction. Intuitively, two CPSs are physically-disjoint if they have different plants but they may share logical channels for communication purposes. More precisely, physically-disjoint CPSs have disjoint state variables and disjoint physical devices (sensors and actuators). As we consider only well-formed CPSs (Definition 5), this ensures that a CPS cannot physically interfere with a parallel CPS by acting on its physical devices.

Formally, let $S_i = \langle \xi^i_{\mathrm{x}}, \xi^i_{\mathrm{s}}, \xi^i_{\mathrm{a}} \rangle$ and $E_i = \langle evol^i, meas^i, inv^i, safe^i, \xi^i_{\mathrm{w}}, \xi^i_{\mathrm{e}} \rangle$ be physical states and physical environments, respectively, associated to sets of state variables $\mathcal{X}_i$, sets of sensors $\mathcal{S}_i$, and sets of actuators $\mathcal{A}_i$, for $i \in \{1, 2\}$. For $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$ and $\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$, we define:

- the *disjoint union* of the physical states S_1 and S_2, written $S_1 \uplus S_2$, to be the physical state $\langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle$ such that: $\xi_{\mathrm{x}} = \xi^1_{\mathrm{x}} \cup \xi^2_{\mathrm{x}}$, $\xi_{\mathrm{s}} = \xi^1_{\mathrm{s}} \cup \xi^2_{\mathrm{s}}$, and $\xi_{\mathrm{a}} = \xi^1_{\mathrm{a}} \cup \xi^2_{\mathrm{a}}$;
- the *disjoint union* of the physical environments E_1 and E_2, written $E_1 \uplus E_2$, to be the physical environment $\langle evol, meas, inv, safe, \xi_{\mathrm{w}}, \xi_{\mathrm{e}} \rangle$ such that:
  1. $evol = evol^1 \cup evol^2$
  2. $meas = meas^1 \cup meas^2$
  3. $S_1 \uplus S_2 \in inv$ iff $S_1 \in inv^1$ and $S_2 \in inv^2$
  4. $S_1 \uplus S_2 \in safe$ iff $S_1 \in safe^1$ and $S_2 \in safe^2$
  5. $\xi_{\mathrm{w}} = \xi^1_{\mathrm{w}} \cup \xi^2_{\mathrm{w}}$
  6. $\xi_{\mathrm{e}} = \xi^1_{\mathrm{e}} \cup \xi^2_{\mathrm{e}}$.

Definition 8 (Physically-disjoint CPSs)**.** *Let $M_i = E_i; S_i \bowtie P_i$, for $i \in \{1, 2\}$. We say that M_1 and M_2 are* physically-disjoint *if S_1 and S_2 have disjoint sets of state variables, sensors and actuators. In this case, we write $M_1 \uplus M_2$ to denote the CPS defined as $(E_1 \uplus E_2); (S_1 \uplus S_2) \bowtie (P_1 \parallel P_2)$.*

A *pure-logical process* is a process that may interfere on communication channels but it never interferes on physical devices as it never accesses sensors and/or actuators. Basically, a pure-logical process is a TPL process [27]. Thus, in a system $M \parallel Q$, where M is an arbitrary CPS, a pure-logical process Q cannot interfere with the physical evolution of M. A process Q can, however, definitely interact with M via communication channels, and hence affect its observable behaviour.

Definition 9 (Pure-logical processes)**.** *A process P is called* pure-logical *if it never acts on sensors and/or actuators.*

Now, we can finally state the compositionality of our trace preorder $\sqsubseteq$ (the proof can be found in the appendix).

Theorem 1 (Compositionality of $\sqsubseteq$)**.** *Let M and N be two arbitrary CPSs in* `CCPSA`*.*

1. *$M \sqsubseteq N$ implies $M \uplus O \sqsubseteq N \uplus O$, for any physically-disjoint CPS O;*

2. *$M \sqsubseteq N$ implies $M \parallel P \sqsubseteq N \parallel P$, for any pure-logical process P;*

3. *$M \sqsubseteq N$ implies $M\backslash c \sqsubseteq N\backslash c$, for any channel c.*

The reader may wonder whether our trace preorder $\sqsubseteq$ is preserved by more permissive contexts. The answer is no. Suppose that in the second item of Theorem 1 we allowed a process P that can also read on sensors. In this case, even if $M \sqsubseteq N$, the parallel process P might read a different value in the two systems at the very same sensor s (due to the sensor error) and transmit these different values on a free channel, breaking the congruence. Activities on actuators may also lead to different behaviours of the compound systems: M and N may have physical components that are not exactly aligned. A similar reasoning applies when composing CPSs with non physically-disjoint ones: noise on physical devices may break the compositionality result.

As we are interested in formalising timing aspects of attacks, such as beginning and duration, we propose a timed variant of $\sqsubseteq$ up to (a possibly infinite) *discrete time interval* $m..n$, with $m \in \mathbb{N}^+$ and $n \in \mathbb{N}^+ \cup \infty$. Intuitively, we write $M \sqsubseteq_{m..n} N$ if the CPS N simulates the execution traces of M in all time slots, except for those contained in the discrete time interval $m..n$.

Definition 10 (Trace preorder up to a time interval)**.** *We write $M \sqsubseteq_{m..n} N$, for $m \in \mathbb{N}^+$ and $n \in \mathbb{N}^+ \cup \{\infty\}$, with $m \leq n$, if the following conditions hold:*

- *m is the minimum integer for which there is a trace t, with $\#\mathsf{tick}(t)=m-1$, s.t. $M \xrightarrow{t}$ and $N \not\xRightarrow{\hat{t}}$;*
- *n is the infimum element of $\mathbb{N}^+ \cup \{\infty\}$, $n \geq m$, such that whenever $M \xrightarrow{t_1} M'$, with $\#\mathsf{tick}(t_1) = n-1$, there is t_2, with $\#\mathsf{tick}(t_1) = \#\mathsf{tick}(t_2)$, such that $N \xrightarrow{t_2} N'$, for some N', and $M' \sqsubseteq N'$.*

In Definition 10, the first item says that N can simulate the traces of M for at most $m-1$ time slots; whereas the second item says two things: (i) in time interval $m..n$ the simulation does not hold; (ii) starting from the time slot $n+1$ the CPS N can simulate again the traces of M. Note that $\inf(\emptyset) = \infty$. Thus, if $M \sqsubseteq_{m..\infty} N$, then N simulates M only in the first $m-1$ time slots.

Theorem 2 (Compositionality of $\sqsubseteq_{m..n}$)**.** *Let M and N be two arbitrary CPSs in* `CCPSA`*.*

1. *$M \sqsubseteq_{m..n} N$ implies that for any physically-disjoint CPS there are $m', n' \in \mathbb{N}^+ \cup \infty$, with $m'..n' \subseteq m..n$ such that $M \uplus O \sqsubseteq_{m'..n'} N \uplus O$;*

2. *$M \sqsubseteq_{m..n} N$ implies that for any pure-logical process P there are $m', n' \in \mathbb{N}^+ \cup \infty$, with $m'..n' \subseteq m..n$ such that $M \parallel P \sqsubseteq_{m'..n'} N \parallel P$;*

3. *$M \sqsubseteq_{m..n} N$ implies that for any channel c there are $m', n' \in \mathbb{N}^+ \cup \infty$, with $m'..n' \subseteq m..n$ such that $M\backslash c \sqsubseteq_{m'..n'} N\backslash c$.*

The proof can be found in the appendix.

3 A Running Example

In this section, we introduce a running example to illustrate how we can precisely represent CPSs and a variety of different physics-based attacks. In practice, we formalise a relatively simple CPS *Sys* in which the temperature of an *engine* is maintained within a specific range by means of a cooling system. We wish to remark here that while we have kept the example simple, it is actually far from trivial and designed to describe a wide number of attacks. The main structure of the CPS *Sys* is shown in Figure 2.

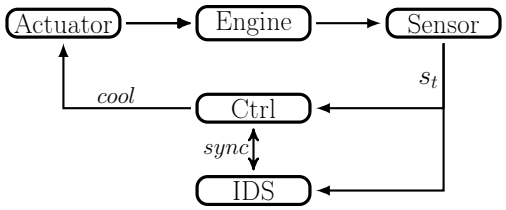

Figure 2: The main structure of the CPS *Sys*

3.1 The CPS *Sys*

The physical state *State* of the engine is characterised by: (i) a state variable *temp* containing the current temperature of the engine, and an integer state variable *stress* keeping track of the level of stress of the mechanical parts of the engine due to high temperatures (exceeding 9.9 degrees); this integer variable ranges from 0, meaning no stress, to 5, for high stress; (ii) a sensor s_{t} (such as a thermometer or a thermocouple) measuring the temperature of the engine, (iii) an actuator *cool* to turn on/off the cooling system.

The physical environment of the engine, *Env*, is constituted by: (i) a simple evolution law *evol* that increases (respectively, decreases) the value of *temp* by one degree per time unit, when the cooling system is inactive (respectively, active), up to the uncertainty of the system; the variable *stress* is increased each time the current temperature is above 9.9 degrees, and dropped to 0 otherwise; (ii) a measurement map *meas* returning the value detected by the sensor s_t, up to the error associated to the sensor; (iii) an invariant set saying that the system gets faulty when the temperature of the engine gets out of the range $[0, 50]$, (iv) a safety set to express that the system moves to an unsafe state when the level of stress reaches the threshold 5, (v) an uncertainty function in which each state variable may evolve with an uncertainty $\delta = 0.4$ degrees, (vi) a sensor-error function saying that the sensor s_{t} has an accuracy $\epsilon = 0.1$ degrees.

Formally, $\mathit{State} = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{s}}, \xi_{\mathrm{a}} \rangle$ where:

- $\xi_{\mathrm{x}} \in \mathbb{R}^{\{\mathit{temp},\mathit{stress}\}}$ and $\xi_{\mathrm{x}}(\mathit{temp}) = 0$ and $\xi_{\mathrm{x}}(\mathit{stress}) = 0$;
- $\xi_{\mathrm{s}} \in \mathbb{R}^{\{s_{\mathrm{t}}\}}$ and $\xi_{\mathrm{s}}(s_{\mathrm{t}}) = 0$;
- $\xi_{\mathrm{a}} \in \mathbb{R}^{\{\mathit{cool}\}}$ and $\xi_{\mathrm{a}}(\mathit{cool}) = \mathsf{off}$; for the sake of simplicity, we can assume ξ_{a} to be a mapping $\{\mathit{cool}\} \to \{\mathsf{on}, \mathsf{off}\}$ such that $\xi_{\mathrm{a}}(\mathit{cool}) = \mathsf{off}$ if $\xi_{\mathrm{a}}(\mathit{cool}) \geq 0$, and $\xi_{\mathrm{a}}(\mathit{cool}) = \mathsf{on}$ if $\xi_{\mathrm{a}}(\mathit{cool}) < 0$;

and $\mathit{Env} = \langle \mathit{evol}, \mathit{meas}, \mathit{inv}, \mathit{safe}, \xi_{\mathrm{w}}, \xi_{\mathrm{e}} \rangle$ with:

- $\mathit{evol}(\xi^i_{\mathrm{x}}, \xi^i_{\mathrm{a}}, \xi_{\mathrm{w}})$ is the set of functions $\xi \in \mathbb{R}^{\{\mathit{temp},\mathit{stress}\}}$ such that:
  - $\xi(\mathit{temp}) = \xi^i_{\mathrm{x}}(\mathit{temp}) + \mathit{heat}(\xi^i_{\mathrm{a}}, \mathit{cool}) + \gamma$, with $\gamma \in [-\delta, +\delta]$ and $\mathit{heat}(\xi^i_{\mathrm{a}}, \mathit{cool}) = -1$ if $\xi^i_{\mathrm{a}}(\mathit{cool}) = \mathsf{on}$ (active cooling), and $\mathit{heat}(\xi^i_{\mathrm{a}}, \mathit{cool}) = +1$ if $\xi^i_{\mathrm{a}}(\mathit{cool}) = \mathsf{off}$ (inactive cooling);
  - $\xi(\mathit{stress}) = \min(5\,,\, \xi^i_{\mathrm{x}}(\mathit{stress})+1)$ if $\xi^i_{\mathrm{x}}(\mathit{temp}) > 9.9$; $\xi(\mathit{stress}) = 0$, otherwise;
- $\mathit{meas}(\xi^i_{\mathrm{x}}, \xi_{\mathrm{e}}) = \big\{\xi : \xi(s_{\mathrm{t}}) \in [\xi^i_{\mathrm{x}}(\mathit{temp}) - \epsilon\,,\, \xi^i_{\mathrm{x}}(\mathit{temp}) + \epsilon]\big\}$;
- $\mathit{inv} = \{\xi^i_{\mathrm{x}} \,:\, 0 \leq \xi^i_{\mathrm{x}}(\mathit{temp}) \leq 50\}$;
- $\mathit{safe} = \{\xi^i_{\mathrm{x}} \,:\, \xi^i_{\mathrm{x}}(\mathit{stress}) < 5\}$ (we recall that the stress threshold is 5);
- $\xi_{\mathrm{w}} \in \mathbb{R}^{\{\mathit{temp},\mathit{stress}\}}$, $\xi_{\mathrm{w}}(\mathit{temp}) = 0.4 = \delta$ and $\xi_{\mathrm{w}}(\mathit{stress}) = 0$;
- $\xi_{\mathrm{e}} \in \mathbb{R}^{\{s_{\mathrm{t}}\}}$ and $\xi_{\mathrm{e}}(s_{\mathrm{t}}) = 0.1 = \epsilon$.

For the cyber component of the CPS *Sys*, we define two parallel processes: *Ctrl* and *IDS*. The former models the *controller* activity, consisting in reading the temperature sensor and in governing the cooling system via its actuator, whereas the latter models a simple *intrusion detection system* that attempts to

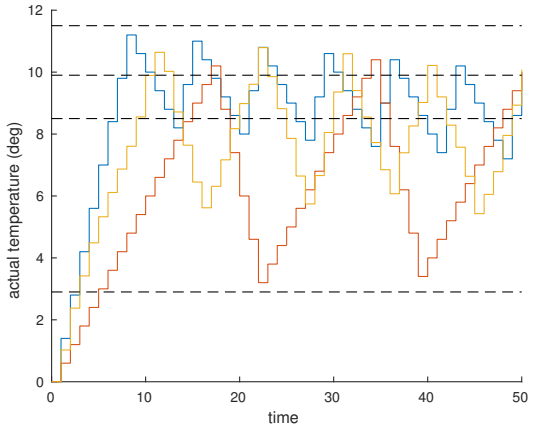

Figure 3: Three possible evolutions of the CPS *Sys*

detect and signal *anomalies* in the behaviour of the system [23]. Intuitively, *Ctrl* senses the temperature of the engine at each time slot. When the *sensed temperature* is above 10 degrees, the controller activates the coolant. The cooling activity is maintained for 5 consecutive time units. After that time, the controller synchronises with the *IDS* component via a private channel *sync*, and then waits for *instructions*, via a channel *ins*. The *IDS* component checks whether the *sensed temperature* is still above 10. If this is the case, it sends an *alarm* of "high temperature", via a specific channel, and then tells *Ctrl* to keep cooling for 5 more time units; otherwise, if the temperature is not above 10, the *IDS* component requires *Ctrl* to stop the cooling activity.

$$
\begin{array}{rcl}
\mathit{Ctrl} & = & \mathsf{read}\ s_{\mathrm{t}}(x).\mathsf{if}\ (x > 10)\ \{\mathit{Cooling}\}\ \mathsf{else}\ \{\mathsf{tick}.\mathit{Ctrl}\} \\
\mathit{Cooling} & = & \mathsf{write}\ \mathit{cool}\langle\mathsf{on}\rangle.\mathsf{tick}^5.\mathit{Check} \\
\mathit{Check} & = & \mathsf{snd}\ \mathit{sync}.\mathsf{rcv}\ \mathit{ins}(y).\mathsf{if}\ (y = \mathsf{keep\_cooling})\ \{\mathsf{tick}^5.\mathit{Check}\}\ \mathsf{else}\ \{\mathsf{write}\ \mathit{cool}\langle\mathsf{off}\rangle.\mathsf{tick}.\mathit{Ctrl}\} \\
\mathit{IDS} & = & \mathsf{rcv}\ \mathit{sync}.\mathsf{read}\ s_{\mathrm{t}}(x).\mathsf{if}\ (x > 10)\ \{\mathsf{snd}\ \mathit{alarm}\langle\mathsf{high\_temp}\rangle.\mathsf{snd}\ \mathit{ins}\langle\mathsf{keep\_cooling}\rangle.\mathsf{tick}.\mathit{IDS}\} \\
 & & \quad \mathsf{else}\ \{\mathsf{snd}\ \mathit{ins}\langle\mathsf{stop}\rangle.\mathsf{tick}.\mathit{IDS}\}\ .
\end{array}
$$

Thus, the whole CPS is defined as:

$$\mathit{Sys} \ = \ \mathit{Env}; \mathit{State} \bowtie (\mathit{Ctrl} \parallel \mathit{IDS}) \backslash \{\mathit{sync}, \mathit{ins}\}$$

For the sake of simplicity, our *IDS* component is quite basic: for instance, it does not check whether the temperature is too low. However, it is straightforward to replace it with a more sophisticated one, containing more informative tests on sensor values and/or on actuators commands.

Figure 3 shows three possible evolutions in time of the state variable *temp* of *Sys*: (i) the first one (in red), in which the temperature of the engine always grows of $1 - \delta = 0.6$ degrees per time unit, when the cooling is off, and always decrease of $1 + \delta = 1.4$ degrees per time unit, when the cooling is on; (ii) the second one (in blue), in which the temperature always grows of $1 + \delta = 1.4$ degrees per time unit, when the cooling is off, and always decreases of $1 - \delta = 0.6$ degrees per time unit, when the cooling is on; (iii) and a third one (in yellow), in which, depending on whether the cooling is off or on, at each time step the temperature grows or decreases of an arbitrary offset lying in the interval $[1 - \delta, 1 + \delta]$.

Our operational semantics allows us to formally prove a number of properties of our running example. For instance, Proposition 1 says that the *Sys* is sound and it never fires the *alarm*.

Proposition 1. *If* $\mathit{Sys} \xrightarrow{t}$ *for some trace* $t = \alpha_1 \dots \alpha_n$*, then* $\alpha_i \in \{\tau, \mathsf{tick}\}$*, for any* $i \in \{1, \dots, n\}$*.*

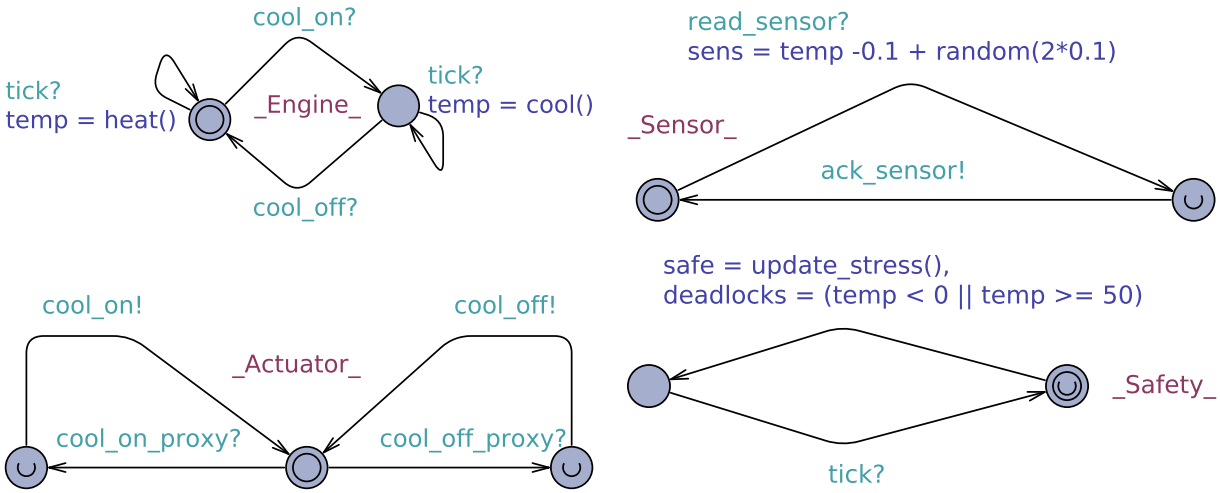

Figure 4: UPPAAL SMC model for the physical component of *Sys*

Actually, we can be quite precise on the temperature reached by *Sys* before and after the cooling: in each of the 5 rounds of cooling, the temperature will drop of a value lying in the real interval $[1-\delta, 1+\delta]$, where δ is the uncertainty.

Proposition 2. *For any execution trace of Sys, we have:*

- *when Sys* turns on *the cooling, the value of the state variable temp ranges over* $(9.9, 11.5]$*;*
- *when Sys* turns off *the cooling, the value of the variable temp ranges over* $(2.9, 8.5]$*.*

The proofs of the Propositions 1 and 2 can be found in the associated technical report [39]. In the following section, we will verify the safety properties stated in these two propositions relying on the statistical model checker UPPAAL SMC [15].

3.2 A formalisation of *Sys* in UPPAAL SMC

In this section, we formalise our running example in UPPAAL SMC [15], the *statistical* extension of the UPPAAL model checker [5] supporting the analysis of systems expressed as composition of *timed and/or probabilistic automata*. In UPPAAL SMC, the user must specify two main statistical parameters α and ϵ, ranging in the interval $[0, 1]$, and representing the probability of *false negatives* and probabilistic *uncertainty*, respectively. Thus, given a CTL property of the system under investigation, the tool returns a probability estimate for that property, lying in a confidence interval $[p - \epsilon, p + \epsilon]$, for some probability $p \in [0, 1]$, with an accuracy $1 - \alpha$. The number of necessary runs to ensure the required accuracy is then computed by the tool relying on the Chernoff-Hoeffding theory [12].

3.2.1 Model

The UPPAAL SMC model of our use case *Sys* is given by three main components represented in terms of *parallel timed automata*: the *physical component*, the *network*, and the *logical component*.

The physical component, whose model is shown in Figure 4, consists of four automata: (i) the *_Engine_* automaton that governs the evolution of the variable *temp* by means of the *heat* and *cool* functions; (ii) the *_Sensor_* automaton that updates the global variable *sens* at each measurement request; (iii) the *_Actuator_* automaton that activates/deactivates the cooling system; (iv) the *_Safety_* automaton that handles the

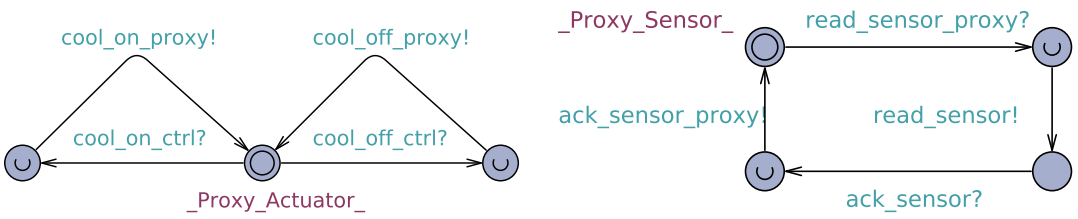

Figure 5: UPPAAL SMC model for the network component of *Sys*

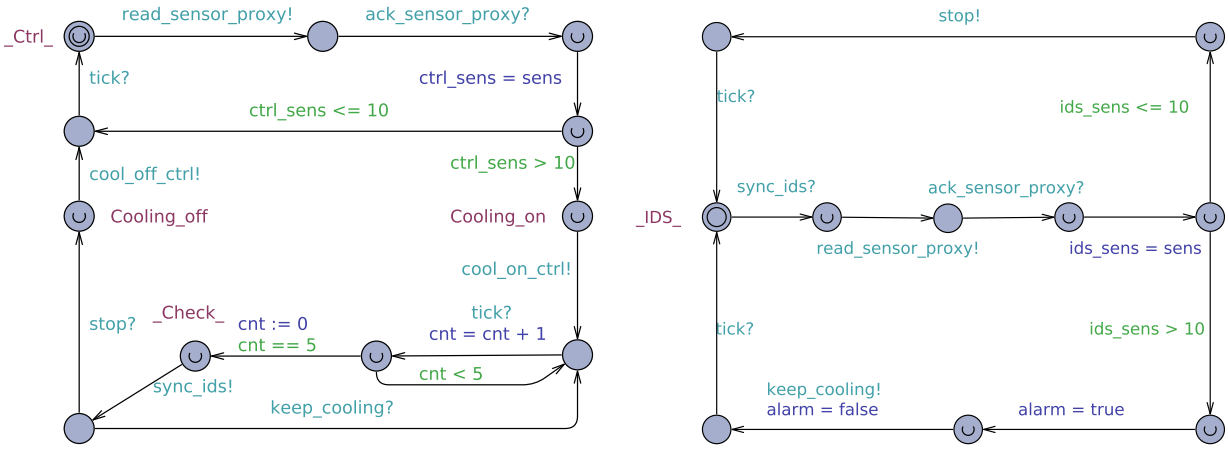

Figure 6: UPPAAL SMC model for the logical component of *Sys*

integer variable *stress*, via the *update_stress* function, and the Boolean variables *safe* and *deadlocks*, associated to the safety set *safe* and the invariant set *inv* of *Sys*, respectively.[5] We also have a small automaton to model a discrete notion of time (via a synchronisation channel tick) as the evolution of state variables is represented via difference equations.

The *network*, whose model is given in Figure 5, consists of *two proxies*: a proxy to relay actuator commands between the actuator device and the controller, a second proxy to relay measurement requests between the sensor device and the logical components (controller and IDS).

The *logical component*, whose model is given in Figure 6, consists of two automata: *_Ctrl_* and *_IDS_* to model the controller and the Intrusion Detection System, respectively; both of them synchronise with their associated proxy copying a fresh value of *sens* into their local variables (*sens_ctrl* and *sens_ids*, respectively). Under proper conditions, the *_IDS_* automaton fires alarms by setting a Boolean variable *alarm*.

3.2.2 Verification

We conduct our safety verification using a notebook with the following set-up: (i) 2.8 GHz Intel i7 7700 HQ, with 16 GB memory, and Linux Ubuntu 16.04 operating system; (ii) UPPAAL SMC model-checker 64-bit, version 4.1.19. The statistical parameters of false negatives (α) and probabilistic uncertainty (ϵ) are both set to 0.01, leading to a confidence level of 99%. As a consequence, having fixed these parameters, for each of our experiments, UPPAAL SMC run a number of runs that may vary from a few hundreds to 26492 (*cf.* Chernoff-Hoeffding bounds).

We basically use UPPAAL SMC to verify properties expressed in terms of *time bounded CTL* formulae of the form $\Box_{[t_1,t_2]} e_{prop}$ and $\Diamond_{[0,t_2]} e_{prop}$[6], where t_1 and t_2 are *time instants* according to the discrete representation

[5] In Section 6.2, we explain why we need to implement an automaton to check for safety conditions rather than verifying a safety property.

[6] The 0 in the left-hand side of the time interval is imposed by the syntax of UPPAAL SMC.

of time in UPPAAL SMC. In practice, we use formulae of the form $\Box_{[t_1,t_2]} e_{prop}$ to compute the probability that a property e_{prop}[7] holds in all *time slots* of the time interval $t_1..t_2$, whereas with formulae of the form $\Diamond_{[0,t_2]} e_{prop}$ we calculate the probability that a property e_{prop} holds in a least one *time slot* of the time interval $0..t_2$.

Thus, instead of proving Proposition 1, we verify, with a 99% accuracy, that in all possible executions that are at most 1000 time slots long, the system *Sys* results to be sound and alarm free, with probability 0.99. Formally, we verify the following three properties:

- $\Box_{[1,1000]}(\neg deadlocks)$, expressing that the system does not deadlock;
- $\Box_{[1,1000]}(safe)$, expressing that the system does not violate the safety conditions;
- $\Box_{[1,1000]}(\neg alarm)$, expressing that the IDS does not fire any alarm.

Furthermore, instead of Proposition 2, we verify, with the same accuracy and for runs of the same length (up to a short initial transitory phase lasting 5 time instants) that if the cooling system is off, then the temperature of the engine lies in the real interval $(2.9, 8.5]$, otherwise it ranges over the interval $(9.9, 11.5]$. Formally, we verify the following two properties:

- $\Box_{[5,1000]}(Cooling\_off \implies (temp > 2.9 \wedge temp \leq 8.5))$
- $\Box_{[5,1000]}(Cooling\_on \implies (temp > 9.9 \wedge temp \leq 11.5))$.

The verification of each of the five properties above requires around 15 minutes. The UPPAAL SMC models of our system and the attacks discussed in the next section are available at the repository `https://bitbucket.org/AndreiMunteanu/cps_smc/src/`.

Remark 3. *In our* UPPAAL *SMC model we decided to represent both uncertainty of physical evolution (in the functions* heat *and* cool *of* _Engine_*) and measurement noise (in* _Sensor_*) in a probabilistic manner via random extractions. Here, the reader may wonder whether it would have been enough to restrict our SMC analysis by considering only upper and lower bounds on these two quantities. Actually, this is not the case because such a restricted analysis might miss admissible execution traces. To see this, suppose to work with a physical uncertainty that is always either* 0.4 *or* -0.4*. Then, the temperature reached by the system would always be of the form* $n.k$*, for* $n, k \in \mathbb{N}$ *and* k *even. As a consequence, our analysis would miss all execution traces in which the system reaches the maximum admissible temperature of* 11.5 *degrees.*

4 Physics-based Attacks

In this section, we use `CCPSA` to formalise a *threat model* of physics-based attacks, i.e., attacks that can manipulate sensor and/or actuator signals in order to drive a *sound* CPS into an undesired state [55]. An attack may have different levels of access to physical devices; for example, it might be able to get read access to the sensors but not write access; or it might get write-only access to the actuators but not read-access. This level of granularity is very important to model precisely how physics-based attacks can affect a CPS [13]. In `CCPSA`, we have a syntactic way to distinguish malicious processes from honest ones.

Definition 11 (Honest system)**.** *A CPS* $E; S \bowtie P$ *is* honest *if* P *is honest, where* P *is honest if it does not contain constructs of the form* $\lfloor \mu.P_1 \rfloor P_2$*.*

We group physics-based attacks in classes that describe both the malicious activities and the timing aspects of the attack. Intuitively, a class of attacks provides information about which physical devices are accessed by the attacks of that class, how they are accessed (read and/or write), when the attack begins and when the attack ends. Thus, let $\mathcal{I}$ be the set of all possible malicious activities on the physical devices of a system, $m \in \mathbb{N}^+$ be the time slot when an attack starts, and $n \in \mathbb{N}^+ \cup \{\infty\}$ be the time slot when the attack ends. We then say that an *attack* A *is of class* $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$ if:

[7] e_{prop} is a side-effect free expression over variables (e.g., clock variables, location names and primitive variables) [5].

1. all possible malicious activities of A coincide with those contained in $\mathcal{I}$;
2. the first of those activities may occur in the m^{th} time slot but not before;
3. the last of those activities may occur in the n^{th} time slot but not after;
4. for $\iota \in \mathcal{I}$, $C(\iota)$ returns a (possibly empty) set of time slots when A may read/tamper with the device ι (this set is contained in $m..n$);
5. C is a total function, i.e., if no attacks of class C can achieve the malicious activity $\iota \in \mathcal{I}$, then $C(\iota) = \emptyset$.

Definition 12 (Class of attacks)**.** *Let $\mathcal{I} = \{\lightning p? : p \in \mathcal{S} \cup \mathcal{A}\} \cup \{\lightning p! : p \in \mathcal{S} \cup \mathcal{A}\}$ be the set of all possible* malicious activities *on physical devices. Let $m \in \mathbb{N}^+$, $n \in \mathbb{N}^+ \cup \{\infty\}$, with $m \leq n$. A* class of attacks *$C \in [\mathcal{I} \to \mathcal{P}(m..n)]$ is a total function such that for any attack A of class C we have:*

(i) $C(\iota) = \{k : A \xrightarrow{t} \xrightarrow{\iota v} A' \wedge k = \#\mathsf{tick}(t) + 1\}$, *for* $\iota \in \mathcal{I}$,

(ii) $m = \inf\{\, k \,:\, k \in C(\iota) \wedge \iota \in \mathcal{I}\,\}$,

(iii) $n = \sup\{\, k \,:\, k \in C(\iota) \wedge \iota \in \mathcal{I}\,\}$.

Along the lines of [17], we can say that an attack A affects a *sound* CPS M if the execution of the compound system $M \parallel A$ differs from that of the original system M, in an observable manner. Basically, a physics-based attack can influence the system under attack in at least two different ways:

- The system $M \parallel A$ might deadlock when M may not; this means that the attack A affects the *availability* of the system. We recall that in the context of CPSs, deadlock is a particular severe physical event.
- The system $M \parallel A$ might have non-genuine execution traces containing observables (violations of safety conditions or communications on channels) that can't be reproduced by M; here the attack affects the *integrity* of the system behaviour.

Definition 13 (Attack tolerance/vulnerability)**.** *Let M be an honest and sound CPS. We say that M is* tolerant to an attack *A if $M \parallel A \sqsubseteq M$. We say that M is* vulnerable to an attack *A if there is a time interval $m..n$, with $m \in \mathbb{N}^+$ and $n \in \mathbb{N}^+ \cup \{\infty\}$, such that $M \parallel A \not\sqsubseteq_{m..n} M$.*

Thus, if a system M is vulnerable to an attack A of class $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$, during the time interval $m'..n'$, then the attack operates during the interval $m..n$ but it influences the system under attack in the time interval $m'..n'$ (obviously, $m' \geq m$). If n' is finite, then we have a *temporary attack*, otherwise we have a *permanent attack*. Furthermore, if $m' - n$ is big enough and $n - m$ is small, then we have a quick nasty attack that affects the system late enough to allow *attack camouflages* [24]. On the other hand, if m' is significantly smaller than n, then the attack affects the observable behaviour of the system well before its termination and the CPS has good chances of undertaking countermeasures to stop the attack. Finally, if $M \parallel A \xrightarrow{t} \xrightarrow{\mathsf{deadlock}}$, for some trace t, then we say that the attack A is *lethal*, as it is capable to halt (deadlock) the CPS M. This is obviously a permanent attack.

Note that, according to Definition 13, the tolerance (or vulnerability) of a CPS also depends on the capability of the *IDS* component to detect and signal undesired physical behaviours. In fact, the *IDS* component might be designed to detect abnormal physical behaviours going well further than deadlocks and violations of safety conditions.

According to the literature, we say that an attack is *stealthy* if it is able to drive the CPS under attack into an incorrect physical state (either deadlock or violation of the safety conditions) without being noticed by the *IDS* component.

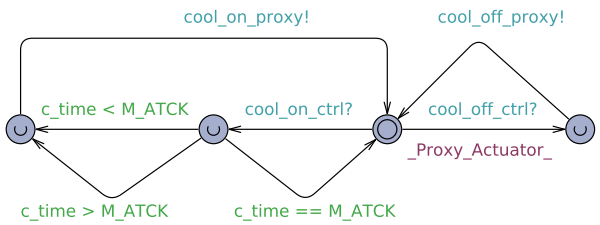

Figure 7: UPPAAL SMC model for the attacker A_m of Example 1

4.1 Three different attacks on the physical devices of the CPS *Sys*

In this subsection, we present three different attacks to the CPS *Sys* described in Section 3. The formal proofs of the propositions stating the tolerance and/or the vulnerability of *Sys* with respect to these three attacks can be found in the associated technical report [39]. Here, we use UPPAAL SMC to verify the models associated to the system under attack in order to detect deadlocks, violations of safety conditions, and IDS failures.

Example 1. *Consider the following* DoS/Integrity attack *on the the actuator cool, of class* $C \in [\mathcal{I} \to \mathcal{P}(m..m)]$ *with* $C(\lightning cool?) = C(\lightning cool!) = \{m\}$ *and* $C(\iota) = \emptyset$*, for* $\iota \notin \{\lightning cool?, \lightning cool!\}$*:*

$$A_m \;=\; \mathsf{tick}^{m-1}.\lfloor \mathsf{drop}\, cool(x).\mathsf{if}\ (x{=}\mathsf{off})\ \{\mathsf{forge}\, cool\langle \mathsf{off}\rangle\}\ \mathsf{else}\ \{\mathsf{nil}\}\rfloor\,.$$

Here, the attack A_m operates exclusively in the m^{th} time slot*, when it tries to drop an eventual cooling command (on or off) coming from the controller, and fabricates a fake command to turn off the cooling system. Thus, if the controller sends in the* m^{th} *time slot a command to turn off the coolant, then nothing bad happens as the attack will put the same message back. On the hand, if the controller sends a command to turn the cooling on, then the attack will drop the command. We recall that the controller will turn on the cooling only if the sensed temperature is greater than* 10 *(and hence temp* > 9.9*); this may happen only if* $m > 8$*. Since the command to turn the cooling on is never re-sent by Ctrl, the temperature will continue to rise, and after only* 4 *time units the system may violate the safety conditions emitting an action* unsafe*, while the IDS component will start sending alarms every* 5 *time units, until the whole system deadlocks because the temperature reaches the threshold of* 50 *degrees. Here, the IDS component of Sys is able to detect the attack with only one time unit delay.*

Proposition 3. *Let Sys be our running example and* A_m *be the attack defined in Example 1. Then,*

- $Sys \parallel A_m \;\sqsubseteq\; Sys$*, for* $1 \leq m \leq 8$*,*
- $Sys \parallel A_m \;\sqsubseteq_{m+4..\infty}\; Sys$*, for* $m > 8$*.*

In order to support the statement of Proposition 3 we verify our UPPAAL SMC model of *Sys* in which the communication network used by the controller to access the actuator is compromised. More precisely, we replace the $\_Proxy\_Actuator\_$ automaton of Figure 5 with a compromised one, provided in Figure 7, that implements the malicious activities of the MITM attacker A_m of Example 1.

We have done our analysis, with a 99% accuracy, for execution traces that are at most 1000 time units long and restricting the *attack time* m in the time interval 1..300. The results of our analysis are:

- when $m \in 1..8$, the attack is harmless as the system results to be safe, deadlock free and alarm free, with probability 0.99;
- when $m \in 9..300$, we have the following situation:
  - the probability that at the attack time m the controller sends a command to activate the cooling system (thus, triggering the attacker that will drop the command) can be obtained by verifying the property $\Diamond_{[0,m]}(Cooling\_on \wedge global\_clock \geq m)$; as shown in Figure 8, when m grows in the time interval 1..300, the resulting probability stabilises around the value 0.096;

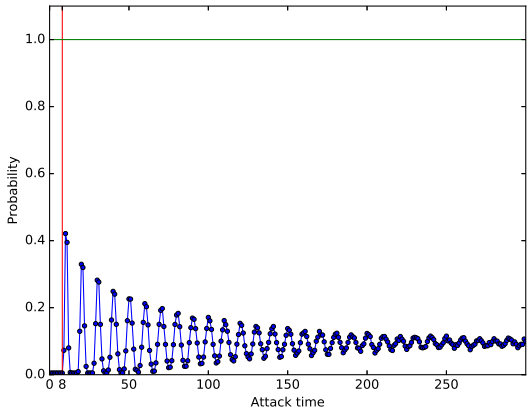

Figure 8: Probability results of $\Diamond_{[0,m]}(\mathit{Cooling\_on} \wedge \mathit{global\_clock} \geq m)$ by varying m in 1..300

- up to the $m{+}3^{\text{th}}$ time slot the system under attack remains safe, i.e., both properties $\Box_{[1,m+3]}(\mathit{safe})$ and $\Box_{[1,m+3]}(\neg \mathit{deadlock})$ hold with probability 0.99;
- up to the $m{+}4^{\text{th}}$ time slot no alarms are fired, i.e., the property $\Box_{[1,m+4]}(\neg \mathit{alarm})$ holds with probability 0.99 (no false positives);
- in the $m{+}4^{\text{th}}$ time slot the system under attack might become unsafe as the probability, for $m \in 9..300$, that the property $\Diamond_{[0,m+4]}(\neg \mathit{safe})$ is satisfied stabilises around the value 0.095;[8]
- in the $m{+}5^{\text{th}}$ time slot the IDS may fire an alarm as the probability, for $m \in 9..300$, that the property $\Diamond_{[0,m+5]}(\mathit{alarm})$ is satisfied stabilises around the value 0.094;[9]
- the system under attack may deadlock as the property $\Diamond_{[0,1000]}(\mathit{deadlocks})$ is satisfied with probability 0.096.[10]

Example 2. *Consider the following* DoS/Integrity attack *to the sensor* s_{t}*, of class* $C \in [\mathcal{I} \to \mathcal{P}(2..\infty)]$ *such that* $C(\lightning s_{\mathrm{t}}?) = \{2\}$*,* $C(\lightning s_{\mathrm{t}}!) = 2..\infty$ *and* $C(\iota) = \emptyset$*, for* $\iota \notin \{\lightning s_{\mathrm{t}}!, \lightning s_{\mathrm{t}}?\}$*. The attack* begins *is activity in the time slot* m*, with* $m > 8$*, and then never stops:*

$$\begin{array}{rcl} A_m & = & \mathsf{tick}^{m-1}.A \\ A & = & \lfloor \mathsf{sniff}\ s_{\mathrm{t}}(x).\mathsf{if}\ (x \leq 10)\ \{B\langle x\rangle\}\ \mathsf{else}\ \{\mathsf{tick}.A\} \rfloor \\ B(y) & = & \lfloor \mathsf{forge}\ s_{\mathrm{t}}\langle y\rangle.\mathsf{tick}.B\langle y\rangle \rfloor B\langle y\rangle\,. \end{array}$$

Here, the attack A_m *behaves as follows. It sleeps for* $m-1$ *time slots and then, in the following time slot, it sniffs the current temperature at sensor* s_{t}*. If the sensed temperature* v *is greater than* 10*, then it moves to the next time slot and restarts sniffing; otherwise from that time on it will keep sending the same temperature* v *to the logical components (controller and IDS). Actually, once the forgery activity starts, the process Ctrl will always receive a temperature below* 10 *and will never activate the cooling system (and consequently the IDS). As a consequence, the system under attack Sys* $\|$ *A will first move to an unsafe state until the invariant*

[8] Since this probability coincides with that of $\Diamond_{[0,m]}(\mathit{Cooling\_on} \wedge \mathit{global\_clock} \geq m)$, it appears very likely that the activation of the cooling system in the m^{th} time slot triggers the attacker whose activity drags the system into an unsafe state with a delay of 4 time slots.

[9] As the two probabilities are pretty much the same, and $\Box_{[1,m+3]}(\mathit{safe})$ and $\Box_{[1,m+4]}(\neg \mathit{alarm})$ hold, the IDS seems to be quite effective in detecting the violations of the safety conditions in the $m{+}4^{\text{th}}$ time slot, with only one time slot delay.

[10] Since the probabilities are still the same, we argue that when the system reaches an unsafe state then it is not able to recover and it is doomed to deadlock.

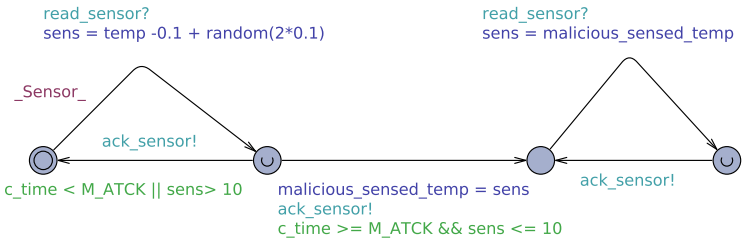

Figure 9: UPPAAL SMC model for the attacker A_m of Example 2

will be violated and the system will deadlock. Indeed, in the worst execution scenario, already in the $m{+}1^{\text{th}}$ *time slot the temperature may exceed* 10 *degrees, and after* 4 tick*-actions, in the* $m{+}5^{\text{th}}$ *time slot, the system may violate the safety conditions emitting an* unsafe *action. Since the temperature will keep growing without any cooling activity, the deadlock of the CPS cannot be avoided. This is a* lethal *attack, as it causes a shut down of the system; it is also a* stealthy attack *as it remains undetected because the IDS never gets into action.*

Proposition 4. *Let Sys be our running example and* A_m*, for* $m > 8$*, be the attack defined in Example 2. Then* $Sys \parallel A_m \sqsubseteq_{m+5..\infty} Sys$*.*

Here, we verify the UPPAAL SMC model of *Sys* in which we assume that its sensor device is compromised (we recall that our MITM forgery attack on sensors or actuators can be assimilated to device compromise). In particular, we replace the *_Sensor_* automaton of Figure 4 with a compromised one, provided in Figure 9, and implementing the malicious activities of the MITM attacker A_m of Example 2.

We have done our analysis, with a 99% accuracy, for execution traces that are at most 1000 time units long and restricting the *attack time* m in the integer interval 9..300. The results of our analysis are:

- up to the $m{+}4^{\text{th}}$ time slot the system under attack remains safe, deadlock free, and alarm free, i.e., all three properties $\Box_{[1,m+4]}(safe)$, $\Box_{[1,m+4]}(\neg deadlock)$, and $\Box_{[1,m+4]}(\neg alarm)$ hold with probability 0.99;
- in the $m{+}5^{\text{th}}$ time slot the system under attack might become unsafe as the probability, for $m \in 9..300$, that the property $\Diamond_{[0,m+5]}(\neg safe)$ is satisfied stabilises around 0.104;
- the system under attack will eventually deadlock not later that 80 time slots after the attack time m, as the property $\Box_{[m+80,1000]}(deadlocks)$ is satisfied with probability 0.99;
- finally, the attack is stealthy as the property $\Box_{[1,1000]}(\neg alarm)$ holds with probability 0.99.

Now, let us examine a similar but less severe attack.

Example 3. *Consider the following DoS/Integrity attack to sensor* s_{t}*, of class* $C \in [\mathcal{I} \to \mathcal{P}(1..n)]$*, with* $C(\lightning s_{\text{t}}!) = C(\lightning s_{\text{t}}?) = 1..n$ *and* $C(\iota) = \emptyset$*, for* $\iota \notin \{\lightning s_{\text{t}}!, \lightning s_{\text{t}}?\}$*:*

$$\begin{array}{rcl} A_n & = & \lfloor \mathsf{sniff}\ s_{\text{t}}(x).\lfloor \mathsf{forge}\ s_{\text{t}}\langle x{-}4\rangle.\mathsf{tick}.A_{n-1} \rfloor A_{n-1} \rfloor A_{n-1}, \quad \text{for } n > 0 \\ A_0 & = & \mathsf{nil}\,. \end{array}$$

In this attack, for n *consecutive time slots,* A_n *sends to the logical components (controller and IDS) the current sensed temperature decreased by an offset* 4*. The effect of this attack on the system depends on the* duration n *of the attack itself: (i) for* $n \leq 8$*, the attack is harmless as the variable temp may not reach a (critical) temperature above* 9.9*; (ii) for* $n = 9$*, the variable temp might reach a temperature above* 9.9 *in the* 9^{th} *time slot, and the attack would delay the activation of the cooling system of one time slot; as a consequence, the system might get into an unsafe state in the time interval* 14..15*, but no alarm will be fired; (iii) for* $n \geq 10$*, the system may get into an unsafe state in the time slot* 14 *and in the following* $n + 11$ *time slots; in this case, this would not be* stealthy attack *as the IDS will fire the alarm with a delay of at most two time slots later, rather this is a* temporary attack *that ends in the time slot* $n + 11$*.*

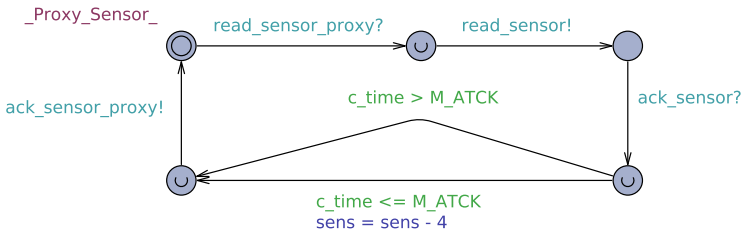

Figure 10: UPPAAL SMC model for the attacker A_n of Example 3

Proposition 5. *Let Sys be our use case and A_n be the attack defined in Example 3. Then:*

- $Sys \parallel A_n \sqsubseteq Sys$, *for* $n \leq 8$,
- $Sys \parallel A_n \sqsubseteq_{14..15} Sys$, *for* $n = 9$,
- $Sys \parallel A_n \sqsubseteq_{14..n+11} Sys$, *for* $n \geq 10$.

Here, we verify the UPPAAL SMC model of *Sys* in which we replace the *_Proxy_Sensor_* automaton of Figure 5 with a compromised one, provided in Figure 10, and implementing the MITM activities of the attacker A_n of Example 3.

We have done our analysis, with a 99% accuracy, for execution traces that are at most 1000 time units long, and assuming that the *duration of the attack* n may vary in the integer interval 1..300. The results of our analysis are:

- when $n \in 1..8$, the system under attack remains safe, deadlock free, and alarm free, i.e., all three properties $\Box_{[1,1000]}(safe)$, $\Box_{[1,1000]}(\neg deadlock)$, and $\Box_{[1,1000]}(\neg alarm)$ hold with probability 0.99;
- when $n = 9$, we have the following situation:
  - the system under attack is deadlock free, i.e., the property $\Box_{[1,1000]}(\neg deadlock)$ holds with probability 0.99;
  - the system remains safe and alarm free, except for the time interval 14..15, i.e., all the following properties $\Box_{[1,13]}(safe)$, $\Box_{[1,13]}(\neg alarm)$, $\Box_{[16,1000]}(safe)$, and $\Box_{[16,1000]}(\neg alarm)$ hold with probability 0.99;
  - in the time interval 14..15, we may have violations of safety conditions, as the property $\Diamond_{[0,14]}(\neg safe \wedge global\_clock \geq 14)$ is satisfied with a probability 0.62, while the property $\Diamond_{[0,15]}(\neg safe \wedge global\_clock \geq 15)$ is satisfied with probability 0.21; both violations are *stealthy* as the property $\Box_{[14,15]}(\neg alarm)$ holds with probability 0.99;
- when $n \geq 10$, we have the following situation:
  - the system is deadlock free, i.e., the property $\Box_{[1,1000]}(\neg deadlock)$ holds with probability 0.99;
  - the system remains safe except for the time interval 14..n+11, i.e., the two properties $\Box_{[1,13]}(safe)$ and $\Box_{[n+12,1000]}(safe)$ hold with probability 0.99;
  - the system is alarm free except for the time interval n+1..n+11, i.e., the two properties $\Box_{[0,n]}(\neg alarm)$ and $\Box_{[n+12,1000]}(\neg alarm)$ hold with probability 0.99;
  - in the 14$^{\text{th}}$ time slot the system under attack may reach an unsafe state as the probability, for $n \in 10..300$, that the property $\Diamond_{[0,14]}(\neg safe \wedge global\_clock \geq 14)$ is satisfied stabilises around 0.548;
  - once the attack has terminated, in the time interval n+1..n+11, the system under attack has good chances to reach an unsafe state as the probability, for $n \in 10..300$, that the property $\Diamond_{[0,n+11]}(\neg safe \wedge n{+}1 \leq global\_clock \leq n{+}11)$ is satisfied stabilises around 0.672;

– the violations of the safety conditions remain completely stealthy only up to the duration n of the attack (we recall that $\Box_{[0,n]}(\neg alarm)$ is satisfied with probability 0.99); the probability, for $n \in 10..300$, that the property $\Diamond_{[0,n+11]}(alarm)$ is satisfied stabilises around 0.13; thus, in the time interval $n{+}1..n{+}11$, only a small portion of violations of safety conditions are detected by the IDS while a great majority of them remains stealthy.

4.2 A technique for proving attack tolerance/vulnerability

In this subsection, we provide sufficient criteria to prove attack tolerance/vulnerability to attacks of an arbitrary class C. Actually, we do more than that: we provide sufficient criteria to prove attack tolerance/vulnerability to all attacks of any class C' that is somehow "weaker" than a given class C.

Definition 14. *Let $C_i \in [\mathcal{I} \to \mathcal{P}(m_i..n_i)]$, for $i \in \{1,2\}$, be two classes of attacks, with $m_1..n_1 \subseteq m_2..n_2$. We say that C_1 is* weaker *than C_2, written $C_1 \preceq C_2$, if $C_1(\iota) \subseteq C_2(\iota)$ for any $\iota \in \mathcal{I}$.*

Intuitively, if $C_1 \preceq C_2$ then: (i) the attacks of class C_1 might achieve fewer malicious activities than any attack of class C_2 (formally, there may be $\iota \in \mathcal{I}$ such that $C_1(\iota) = \emptyset$ and $C_2(\iota) \neq \emptyset$); (ii) for those malicious activities $\iota \in \mathcal{I}$ achieved by the attacks of both classes C_1 and C_2 (i.e., $C_1(\iota) \neq \emptyset$ and $C_2(\iota) \neq \emptyset$), if they may be perpetrated by the attacks of class C_1 at some time slot $k \in m_1..n_1$ (i.e., $k \in C_1(\iota)$) then all attacks of class C_2 may do the same activity ι at the same time k (i.e., $k \in C_2(\iota)$).

The next objective is to define a notion of *most powerful attack* (also called *top attacker*) of a given class C, such that, if a CPS M tolerates the most powerful attack of class C then it also tolerates *any* attack of class C', with $C' \preceq C$. We will provide a similar condition for attack vulnerability: let M be a CPS vulnerable to $Top(C)$ in the time interval $m_1..n_1$; then, for any attack A of class C', with $C' \preceq C$, if M is vulnerable to A then it is so for a smaller time interval $m_2..n_2 \subseteq m_1..n_1$.

Our notion of top attacker has two extra ingredients with respect to the physics-based attacks seen up to now: (i) *nondeterminism*, and (ii) time-unguarded recursive processes. This extra power of the top attacker is not a problem as we are looking for sufficient criteria.

With respect to nondeterminism, we assume a generic procedure $rnd()$ that given an arbitrary set $\mathcal{Z}$ returns an element of $\mathcal{Z}$ chosen in a nondeterministic manner. This procedure allows us to express *nondeterministic choice*, $P \oplus Q$, as an abbreviation for the process $\mathsf{if}\ (rnd(\{\mathsf{true}, \mathsf{false}\}))\ \{P\}\ \mathsf{else}\ \{Q\}$. Thus, let $\iota \in \{\lightning p? : p \in \mathcal{S} \cup \mathcal{A}\} \cup \{\lightning p! : p \in \mathcal{S} \cup \mathcal{A}\}$, $m \in \mathbb{N}^+$, $n \in \mathbb{N}^+ \cup \{\infty\}$, with $m \leq n$, and $\mathcal{T} \subseteq m..n$, we define the attack process $Att(\iota, k, \mathcal{T})$[11] as the attack which may achieve the malicious activity ι, at the time slot k, and which tries to do the same in all subsequent time slots of $\mathcal{T}$. Formally,

$$
\begin{aligned}
Att(\lightning a?, k, \mathcal{T}) = \mathsf{if}\ (k \in \mathcal{T})\ \{(\lfloor \mathsf{drop}\, a(x).Att(\lightning a?, k, \mathcal{T}) \rfloor Att(\lightning a?, k{+}1, \mathcal{T})) \oplus \mathsf{tick}.Att(\lightning a?, k{+}1, \mathcal{T})\} \\
\mathsf{else}\ \{\mathsf{if}\ (k < \sup(\mathcal{T}))\ \{\mathsf{tick}.Att(\lightning a?, k{+}1, \mathcal{T})\}\ \mathsf{else}\ \{\mathsf{nil}\}\} \\
Att(\lightning s?, k, \mathcal{T}) = \mathsf{if}\ (k \in \mathcal{T})\ \{(\lfloor \mathsf{sniff}\, s(x).Att(\lightning s?, k, \mathcal{T}) \rfloor Att(\lightning s?, k{+}1, \mathcal{T})) \oplus \mathsf{tick}.Att(\lightning s?, k{+}1, \mathcal{T})\} \\
\mathsf{else}\ \{\mathsf{if}\ (k < \sup(\mathcal{T}))\ \{\mathsf{tick}.Att(\lightning s?, k{+}1, \mathcal{T})\}\ \mathsf{else}\ \{\mathsf{nil}\}\} \\
Att(\lightning p!, k, \mathcal{T}) = \mathsf{if}\ (k \in \mathcal{T})\ \{(\lfloor \mathsf{forge}\, p\langle rnd(\mathbb{R})\rangle.Att(\lightning p!, k, \mathcal{T}) \rfloor Att(\lightning p!, k{+}1, \mathcal{T})) \oplus \mathsf{tick}.Att(\lightning p!, k{+}1, \mathcal{T})\} \\
\mathsf{else}\ \{\mathsf{if}\ (k < \sup(\mathcal{T}))\ \{\mathsf{tick}.Att(\lightning p!, k{+}1, \mathcal{T})\}\ \mathsf{else}\ \{\mathsf{nil}\}\}\,.
\end{aligned}
$$

Note that, for $\mathcal{T} = \emptyset$, we assume $\sup(\mathcal{T}) = -\infty$. We can now use the definition above to formalise the notion of most powerful attack of a given class C.

Definition 15 (Top attacker)**.** *Let $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$ be a class of attacks. We define*

$$Top(C) \;=\; \prod_{\iota \in \mathcal{I}} Att(\iota, 1, C(\iota))$$

as the most powerful attack*, or* top attacker*, of class C.*

[11] In case of sensor sniffing, we might avoid to add this specific attack process as our top attacker process can forge any possible value without need to read sensors.

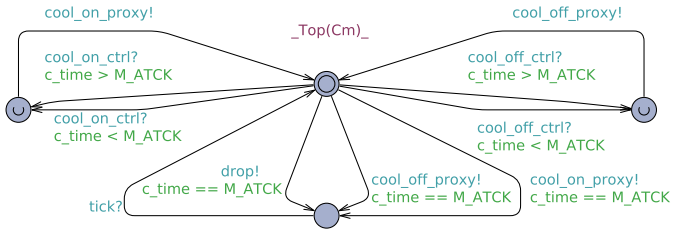

Figure 11: UPPAAL SMC model for the top attacker $Top(C_m)$ of Example 4

The following theorem provides soundness criteria for attack tolerance and attack vulnerability.

Theorem 3 (Soundness criteria)**.** *Let M be an honest and sound CPS, C an arbitrary class of attacks, and A an attack of a class C', with $C' \preceq C$.*

- *If $M \parallel Top(C) \sqsubseteq M$ then $M \parallel A \sqsubseteq M$.*
- *If $M \parallel Top(C) \sqsubseteq_{m_1..n_1} M$ then either $M \parallel A \sqsubseteq M$ or $M \parallel A \sqsubseteq_{m_2..n_2} M$, with $m_2..n_2 \subseteq m_1..n_1$.*

Corollary 1. *Let M be an honest and sound CPS, and C a class of attacks. If $Top(C)$ is not lethal for M then any attack A of class C', with $C' \preceq C$, is not lethal for M. If $Top(C)$ is not a permanent attack for M, then any attack A of class C', with $C' \preceq C$, is not a permanent attack for M.*

The following example illustrates how Theorem 3 could be used to infer attack tolerance/vulnerability with respect to an entire class of attacks.

Example 4. *Consider our running example Sys and a class of attacks C_m, for $m \in \mathbb{N}$, such that $C_m(\lightning cool?) = C_m(\lightning cool!) = \{m\}$ and $C_m(\iota) = \emptyset$, for $\iota \notin \{\lightning cool?, \lightning cool!\}$. Attacks of class C_m may tamper with the actuator cool only in the time slot m (i.e., in the time interval $m..m$). The attack A_m of Example 1 is of class C_m.*

In the following analysis in UPPAAL SMC of the top attacker $Top(C_m)$, we will show that both the vulnerability window and the probability of successfully attacking the system represent an upper bound for the attack A_m of Example 1 of class C_m. Technically, we verify the UPPAAL SMC model of *Sys* in which we replace the *_Proxy_Actuator_* automaton of Figure 5 with a compromised one, provided in Figure 11, and implementing the activities of the top attacker $Top(C_m)$. We carry out our analysis with a 99% accuracy, for execution traces that are at most 1000 time slots long, limiting the *attack time* m to the integer interval 1..300.

To explain our analysis further, let us provide details on how $Top(C_m)$ affects *Sys* when compared to the attacker A_m of class C_m seen in the Example 1.

- In the time interval $1..m$, the attacked system remains safe, deadlock free, and alarm free. Formally, the three properties $\Box_{[1,m]}(safe)$, $\Box_{[1,m]}(\neg deadlock)$ and $\Box_{[1,m]}(\neg alarm)$ hold with probability 0.99. Thus, in this time interval, the top attacker is harmless, as well as A_m.
- In the time interval $m{+}1..m{+}3$, the system exposed to the top attacker may deadlock when $m \in 1..8$; for $m > 8$ the system under attack is deadlock free (see Figure 12). This is because the top attacker, unlike the attacker A_m, can forge in the first 8 time slots cool-on commands turning on the cooling and dropping the temperature below zero in the time interval $m{+}1..m{+}3$. Note that no alarms or unsafe behaviours occur in this case, as neither the safety process nor the IDS check whether the temperature drops below a certain threshold. Formally, the properties $\Box_{[m+1,m+3]}(safe)$ and $\Box_{[m+1,m+3]}(\neg alarm)$ hold with probability 0.99, as already seen for the attacker A_m.
- In the time interval $m{+}4..1000$, the top attacker has better chances to deadlock the system when compared with the attacker A_m (see Figure 13). With respect to safety and alarms, the top attacker and the attacker A_m have the same probability of success (the properties $\Box_{[m+4,1000]}(safe)$ and $\Box_{[m+4,1000]}(\neg alarm)$ return the same probability results).

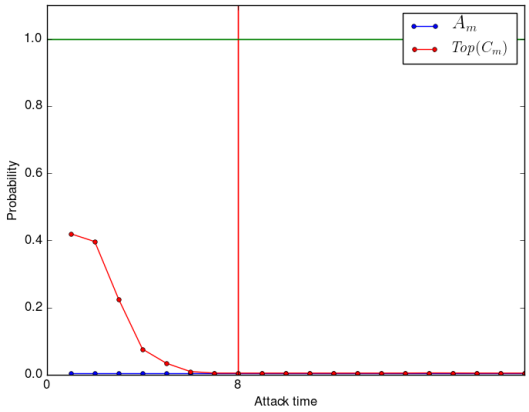

Figure 12: Results of $\Diamond_{[0,m+3]}(deadlock \wedge global\_clock \geq m+1)$ by varying the attack time m

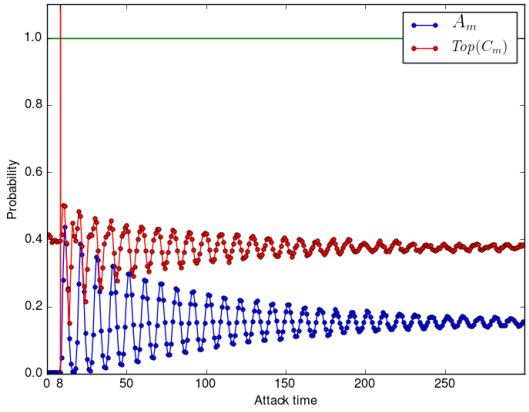

Figure 13: Results of $\Diamond_{[0,1000]}(deadlock \wedge global\_clock \geq m+4)$ by varying the attack time m

This example shows how the verification of a top attacker $Top(C)$ provides an upper bound of the effectiveness of the entire class of attacks C, in terms of both vulnerability window and probability of successfully attack the system. Of course, the accuracy of such approximation cannot be estimated a priori.

5 Impact of a physics-based attack

In the previous section, we have grouped physics-based attacks by focussing on the physical devices under attack and the timing aspects of the attack (Definition 12). Then, we have provided a formalisation of when a CPS should be considered tolerant/vulnerable to an attack (Definition 13). In this section, we show that these two formalisations are important not only to demonstrate the tolerance (or vulnerability) of a CPS with respect to certain attacks, but also to evaluate the disruptive impact of those attacks on the target CPS [21, 44].

The goal of this section is to provide a *formal metric* to estimate the impact of a *successful attack* on the *physical behaviour* of a CPS. In particular, we focus on the ability that an attack may have to drag a CPS out of the correct behaviour modelled by its evolution map, with the given uncertainty.

Recall that *evol* is *monotone* with respect to the uncertainty. Thus, as stated in Proposition 6, an increase of the uncertainty may translate into a widening of the range of the possible behaviours of the CPS. In the following, given the physical environment $E = \langle evol, meas, inv, safe, \xi_{\mathrm{w}}, \xi_{\mathrm{e}} \rangle$, we write $E[\xi_{\mathrm{w}} \leftarrow \xi'_{\mathrm{w}}]$ as an abbreviation for $\langle evol, meas, inv, safe, \xi'_{\mathrm{w}}, \xi_{\mathrm{e}} \rangle$; similarly, for $M = E; S \bowtie P$ we write $M[\xi_{\mathrm{w}} \leftarrow \xi'_{\mathrm{w}}]$ for $E[\xi_{\mathrm{w}} \leftarrow \xi'_{\mathrm{w}}]; S \bowtie P$.

Proposition 6 (Monotonicity)**.** *Let M be an honest and sound CPS with uncertainty ξ_{w}. If $\xi_{\mathrm{w}} \leq \xi'_{\mathrm{w}}$ and $M \xrightarrow{t} M'$ then $M[\xi_{\mathrm{w}} \leftarrow \xi'_{\mathrm{w}}] \xrightarrow{t} M'[\xi_{\mathrm{w}} \leftarrow \xi'_{\mathrm{w}}]$.*

However, a wider uncertainty in the model does not always correspond to a widening of the possible behaviours of the CPS. In fact, this depends on the *intrinsic tolerance* of a CPS with respect to changes in the uncertainty function. In the following, we will write $\xi_{\mathrm{w}} + \xi'_{\mathrm{w}}$ to denote the function $\xi''_{\mathrm{w}} \in \mathbb{R}^{\mathcal{X}}$ such that $\xi''_{\mathrm{w}}(x) = \xi_{\mathrm{w}}(x) + \xi'_{\mathrm{w}}(x)$, for any $x \in \mathcal{X}$.

Definition 16 (System ξ-tolerance)**.** *An honest and sound CPS M with uncertainty ξ_{w} is said* ξ-tolerant*, for $\xi \in \mathbb{R}^{\mathcal{X}}$ and $\xi \geq 0$, if*

$$\xi \;=\; \sup\big\{\xi' :\; M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}} + \eta] \sqsubseteq M,\ \textit{for any } 0 \leq \eta \leq \xi'\big\}.$$

Intuitively, if a CPS M has been designed with a given uncertainty ξ_{w}, but M is actually ξ-tolerant, with $\xi > 0$, then the uncertainty ξ_{w} is somehow underestimated: the real uncertainty of M is given by $\xi_{\mathrm{w}} + \xi$. This information is quite important when trying to estimate the impact of an attack on a CPS. In fact, if a system M has been designed with a given uncertainty ξ_{w}, but M is actually ξ-tolerant, with $\xi > 0$, then an attack has (at least) a "room for manoeuvre" ξ to degrade the whole CPS without being observed (and hence detected).

Let *Sys* be our running example. In the rest of the section, with an abuse of notation, we will write $Sys[\delta \leftarrow \gamma]$ to denote *Sys* where the uncertainty δ of the variable *temp* has been replaced with γ.

Example 5. *The CPS Sys is $\frac{1}{20}$-tolerant as $\sup\big\{\xi' : Sys[\delta \leftarrow \delta{+}\eta] \sqsubseteq Sys,\ \textit{for } 0 \leq \eta \leq \xi'\big\}$ is equal to $\frac{1}{20}$. Since $\delta + \xi = \frac{8}{20} + \frac{1}{20} = \frac{9}{20}$, then this statement relies on the following proposition whose proof can be found in the associated technical report [39].*

Proposition 7. *We have*

- $Sys[\delta \leftarrow \gamma] \sqsubseteq Sys$, *for* $\gamma \in (\frac{8}{20}, \frac{9}{20})$,
- $Sys[\delta \leftarrow \gamma] \not\sqsubseteq Sys$, *for* $\gamma > \frac{9}{20}$.

Now everything is in place to define our metric to estimate the impact of an attack.

Definition 17 (Impact)**.** *Let M be an honest and sound CPS with uncertainty ξ_{w}. We say that an attack A has* definitive impact ξ *on the system M if*

$$\xi = \inf\big\{\xi' : \xi' \in \mathbb{R}^{\mathcal{X}} \;\wedge\; \xi'{>}0 \;\wedge\; M \parallel A \sqsubseteq M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}} + \xi']\big\}.$$

It has pointwise impact ξ *on the system M at time m if*

$$\xi{=}\inf\big\{\xi' : \xi' \in \mathbb{R}^{\mathcal{X}} \;\wedge\; \xi'{>}0 \;\wedge\; M \parallel A \sqsubseteq_{m..n} M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}} + \xi'], n \in \mathbb{N}^{+} \cup \{\infty\}\big\}.$$

Intuitively, the impact of an attacker A on a system M measures the perturbation introduced by the presence of the attacker in the compound system $M \parallel A$ with respect to the original system M. With this definition, we can establish either the definitive (and hence maximum) impact of the attack A on the system M, or the impact at a specific time m. In the latter case, by definition of $\sqsubseteq_{m..n}$, there are two possibilities: either the impact of the attack keeps growing after time m, or in the time interval $m{+}1$, the system under attack deadlocks.

The impact of $Top(C)$ provides an upper bound for the impact of all attacks of class C', $C' \preceq C$, as shown in the following theorem (proved in the appendix).

Theorem 4 (Top attacker's impact)**.** *Let M be an honest and sound CPS, and C an arbitrary class of attacks. Let A be an arbitrary attack of class C', with $C' \preceq C$.*

- *The definitive impact of $Top(C)$ on M is greater than or equal to the definitive impact of A on M.*
- *If $Top(C)$ has pointwise impact ξ on M at time m, and A has pointwise impact ξ' on M at time m', with $m' \leq m$, then $\xi' \leq \xi$.*

In order to help the intuition on the impact metric defined in Definition 17, we give a couple of examples. Here, we focus on the role played by the size of the vulnerability window.

Example 6. *Let us consider the attack A_n of Example 3, for $n \in \{8, 9, 10\}$. Then,*

- *A_8 has definitive impact 0 on Sys,*
- *A_9 has definitive impact $0.2\overline{3}$ on Sys,*
- *A_{10} has definitive impact 0.4 on Sys.*

Formally, the impacts of these three attacks are obtained by calculating

$$\inf\{\xi' : \xi' > 0 \wedge Sys \parallel A_n \sqsubseteq Sys[\delta \leftarrow \delta + \xi']\},$$

for $n \in \{8, 9, 10\}$. Attack A_9 has a very low impact on Sys as it may drag the system into a temporary unsafe state *in the time interval* $14..15$*, whereas A_{10} has a slightly stronger impact as it may induce a* temporary unsafe state *during the larger time interval* $14..21$*. Technically, since $\delta + \xi = 0.4 + 0.4 = 0.8$, the calculation of the impact of A_{10} relies on the following proposition whose proof can be found in the associated technical report [39].*

Proposition 8. *Let A_{10} be the attack defined in Example 3. Then:*

- *$Sys \parallel A_{10} \not\sqsubseteq Sys[\delta \leftarrow \gamma]$, for $\gamma \in (0.4, 0.8)$,*
- *$Sys \parallel A_{10} \sqsubseteq Sys[\delta \leftarrow \gamma]$, for $\gamma > 0.8$.*

On the other hand, the attack provided in Example 2, driving the system to a (*permanent*) *deadlock state*, has a much stronger impact on the CPS *Sys* than the attack of Example 3.

Example 7. *Let us consider the attack A_m of Example 2, for $m > 8$. As already discussed, this is a stealthy lethal attack that has a very severe and high impact. In fact, it has a definitive impact of 8.5 on the CPS Sys. Formally,*

$$8.5 = \inf\{\, \xi' : \xi' > 0 \wedge Sys \parallel A_m \sqsubseteq Sys[\delta \leftarrow \delta + \xi']\}.$$

Technically, since $\delta + \xi = 0.4 + 8.5 = 8.9$, what stated in this example relies on the following proposition whose proof can be found in the associated technical report [39].

Proposition 9. *Let A_m be the attack defined in Example 2. Then:*

- *$Sys \parallel A_m \not\sqsubseteq Sys[\delta \leftarrow \gamma]$, for $\gamma \in (0.4, 8.9)$,*
- *$Sys \parallel A_m \sqsubseteq Sys[\delta \leftarrow \gamma]$, for $\gamma > 8.9$.*

Thus, Definition 17 provides an instrument to estimate the impact of a *successful attack* on a CPS in terms of the perturbation introduced both on its physical and on its logical processes. However, there is at least another question that a CPS designer could ask: "Is there a way to estimate the chances that an attack will be successful during the execution of my CPS?" To paraphrase in a more operational manner: how many execution traces of my CPS are prone to be attacked by a specific attack? As argued in the future work, we believe that *probabilistic metrics* might reveal to be very useful in this respect [41].

6 Conclusions, related and future work

6.1 Summary

We have provided *theoretical foundations* to reason about and *formally* detect attacks to physical devices of CPSs. A straightforward utilisation of these methodologies is for *model-checking* or *monitoring* in order to be able to formally analyse security properties of CPSs either before system deployment or, when static analysis is not feasible, at runtime to promptly detect undesired behaviours. To that end, we have proposed a hybrid process calculus, called `CCPSA`, as a formal *specification language* to model physical and cyber components of CPSs as well as MITM physics-based attacks. Note that our calculus is general enough to represent *Supervisory Control And Data Acquisition* (SCADA) systems as cyber components which can easily interact with controllers and IDSs via channel communications. SCADA systems are the main technology used by system engineers to supervise the activities of complex CPSs.

Based on `CCPSA` and its labelled transition semantics, we have formalised a threat model for CPSs by grouping physics-based attacks in classes, according to the target physical devices and two timing parameters: begin and duration of the attacks. Then, we developed two different *compositional* trace semantics for `CCPSA` to assess *attack tolerance/vulnerability* with respect to a given attack. Such a tolerance may hold *ad infinitum* or for a limited amount of time. In the latter case, the CPS under attack is vulnerable and the attack affects the observable behaviour of the system only after a certain point in time, when the attack itself may already be achieved or still working.

Along the lines of GNDC [17], we have defined a notion of *top attacker*, $Top(C)$, of a given class of attacks C, which has been used to provide sufficient criteria to prove attack tolerance/vulnerability to all attacks of class C (and weaker ones).

Then, we have provided a metric to estimate the *maximum impact* introduced in the system under attack with respect to its genuine behaviour, according to its evolution law and the uncertainty of the model. We have proved that the impact of the most powerful attack $Top(C)$ represents an upper bound for the impact of any attack A of class C (and weaker ones).

Finally, we have formalised a *running example* in Uppaal SMC [15], the statistical extension of the Uppaal model checker [5]. Our goal was to test Uppaal SMC as an automatic tool for the *static security analysis* of a simple but significant CPS exposed to a number of different physics-based attacks with different impacts on the system under attack. Here, it is important to note that, although we have verified most of the properties stated in the paper, we have not been able to capture time properties on the responsiveness of the IDS to violations of the safety conditions. Examples of such properties are: (i) there are time slots m and k such that the system may have an unsafe state at some time $n > m$, and the IDS detects this violation with a delay of at least k time slots (k being a lower bound of the reaction time of the IDS), or (ii) there is a time slot n in which the IDS fires an alarm but neither an unsafe state nor a deadlock occurs in the time interval $n{-}k..n{+}k$: this would provide a tolerance of the occurrence of *false positive*. Furthermore, Uppaal SMC does not support the verification of nested formulae. Thus, although from a designer's point of view it would have been much more practical to verify a logic formula of the form $\exists\Diamond(\Box_{[t,t+5]} temp > 9.9)$ to check safety and invariant conditions, in Uppaal SMC we had to implement a _Safety_ automaton that is not really part of our CPS (for more details see the discussion of related work).

6.2 Related work

A number of approaches have been proposed for modelling CPSs using *hybrid process algebras* [14, 7, 57, 52, 20]. Among these approaches, our calculus `CCPSA` shares some similarities with the ϕ-calculus [52]. However, unlike `CCPSA`, in the ϕ-calculus, given a hybrid system (E, P), the process P can dynamically change the evolution law in E. Furthermore, the ϕ-calculus does not have a representation of physical devices and measurement law, which are instead crucial for us to model physics-based attacks that operate in a timely fashion on sensors and actuators. More recently, Galpin et al. [20] have proposed a process algebra in which the continuous part of the system is represented by appropriate variables whose changes are determined by active influences (i.e., commands on actuators).

Many good surveys on the security of cyber-physical systems have been published recently (see, e.g., [23, 62, 2, 63]), including a survey of surveys [22]. In particular, the surveys [63, 62] provide a systematic categorisation of 138 selected papers on CPS security. Among those 138 papers, 65 adopt a discrete notion of time similar to ours, 26 a continuous one, 55 a quasi-static time model, and the rest use a hybrid time model. This study encouraged us in adopting a discrete time model for physical processes rather than a continuous one. Still, one might wonder what is actually lost when one adopts a discrete rather than a continuous time model, in particular when the attacker has the possibility to move in a continuous time setting. A continuous time model is, of course, more expressive. For instance, Kanovich et al. [32] identified a novel vulnerability in the context of *cryptographic protocols* for CPSs in which the attacker works in a continuous-time setting to fool discrete-time verifiers. However, we believe that, for *physics-based attacks*, little is lost by adopting a discrete time model. In fact, sensor measurements and actuator commands are elaborated within controllers, which are digital devices with an intrinsic discrete notion of time. In particular, with respect to dropping of actuator commands and forging of sensor measurements, there are no differences between discrete-time and continuous-time attackers given that to achieve those malicious activities the attacker has to synchronise with the controller. Thus, there remain only two potential malicious activities: sensor sniffing and forging of actuator commands. Can a continuous-time attacker, able to carry out these two malicious activities, be more disruptive than a similar attacker adopting a discrete-time model? This would only be the case when dealing with very rare physical processes changing their physical state in an extremely fast way, faster than the controller which is the one dictating the discrete time of the CPS. However, we believe that CPSs of this kind would be hardly controllable as they would pose serious safety issues even in the absence of any attacker.

The survey [23] provides an exhaustive review of papers on physics-based anomaly detection proposing a unified taxonomy, whereas the survey [2] presents the main solutions in the estimation of the consequences of cyber-attacks, attacks modelling and detection, and the development of security architecture (the main types of attacks and threats against CPSs are analysed and grouped in a tree structure).

Huang et al. [30] were among the first to propose *threat models* for CPSs. Along with [33, 34], they stressed the role played by timing parameters on integrity and DoS attacks.

Gollmann et al. [24] discussed possible goals (*equipment damage*, *production damage*, *compliance violation*) and *stages* (*access*, *discovery*, *control*, *damage*, *cleanup*) of physics-based attacks. In this article, we focused on the "damage" stage, where the attacker already has a rough idea of the plant and the control architecture of the target CPS.As we remarked in Section 1, here we focus on an attacker who has already entered the CPS, without considering how the attacker gained access to the system, which could have happened in several ways, for instance by attacking an Internet-accessible controller or one of the communication protocols.

Almost all papers discussed in the surveys mentioned above [63, 23, 2] investigate attacks on CPSs and their protection by relying on *simulation test systems* to validate the results, rather than *formal methodologies*. We are aware of a number of works applying *formal methods* to CPS security, although they apply methods, and most of the time have goals, that are quite different from ours. We discuss the most significant ones on the following.

Burmester et al. [11] employed *hybrid timed automata* to give a threat framework based on the traditional Byzantine faults model for crypto-security. However, as remarked in [55], physics-based attacks and faults have inherently distinct characteristics. Faults are considered as physical events that affect the system behaviour where simultaneous events don't act in a coordinated way, whereas cyber attacks may be performed over a significant number of attack points and in a coordinated way.

In [59], Vigo presented an attack scenario that addresses some of the peculiarities of a cyber-physical adversary, and discussed how this scenario relates to other attack models popular in the security protocol literature. Then, in [60] Vigo et al. proposed an untimed calculus of broadcasting processes equipped with notions of failed and unwanted communication. These works differ quite considerably from ours, e.g., they focus on *DoS attacks* without taking into consideration timing aspects or impact of the attack.

Cómbita et al. [13] and Zhu and Basar [64] applied *game theory* to capture the conflict of goals between an attacker who seeks to maximise the damage inflicted to a CPS's security and a defender who aims to minimise it [43].

Rocchetto and Tippenhauer [51] introduced a taxonomy of the diverse attacker models proposed for CPS

security and outline requirements for generalised attacker models; in [50], they then proposed an extended *Dolev-Yao attacker model* suitable for CPSs. In their approach, physical layer interactions are modelled as abstract interactions between logical components to support reasoning on the physical-layer security of CPSs. This is done by introducing additional orthogonal channels. Time is not represented.

Nigam et al. [46] worked around the notion of *Timed Dolev-Yao Intruder Models for Cyber-Physical Security Protocols* by bounding the number of intruders required for the automated verification of such protocols. Following a tradition in security protocol analysis, they provide an answer to the question: How many intruders are enough for verification and where should they be placed? They also extend the strand space model to CPS protocols by allowing for the symbolic representation of time, so that they can use the tool Maude [47] along with SMT support. Their notion of time is however different from ours, as they focus on the time a message needs to travel from an agent to another. The paper does not mention physical devices, such as sensors and/or actuators.

There are a few approaches that carry out *information flow security analysis* on discrete/continuous models for CPSs. Akella et al. [1] proposed an approach to perform information flow analysis, including both trace-based analysis and automated analysis through process algebra specification. This approach has been used to verify process algebra models of a gas pipeline system and a smart electric power grid system. Bodei et al. [9] proposed a process calculus supporting a control flow analysis that safely approximates the abstract behaviour of IoT systems. Essentially, they track how data spread from sensors to the logics of the network, and how physical data are manipulated. In [8], the same authors extend their work to infer *quantitative measures* to establish the cost of possibly security countermeasures, in terms of time and energy. Another discrete model has been proposed by Wang [61], where Petri-net models have been used to verify *non-deducibility security properties* of a natural gas pipeline system. More recently, Bohrer and Platzer [10] introduced dHL, a hybrid logic for verifying cyber-physical hybrid-dynamic information flows, communicating information through both discrete computation and physical dynamics, so security is ensured even when attackers observe *continuously-changing values* in continuous time.

Huang et al. [29] proposed a *risk assessment method* that uses a Bayesian network to model the attack propagation process and infers the probabilities of sensors and actuators to be compromised. These probabilities are fed into a stochastic hybrid system (SHS) model to predict the evolution of the physical process being controlled. Then, the security risk is quantified by evaluating the system availability with the SHS model.

As regards tools for the formal verification of CPSs, we remark that we tried to verify our case study using *model-checking tools* for distributed systems such as PRISM [36], Uppaal [6], Real-Time Maude [47], and prohver within the Modest Toolset [26]. In particular, as our example adopts a discrete notion of time, we started looking at tools supporting discrete time. PRISM, for instance, relies on Markov decision processes or discrete-time Markov chains, depending on whether one is interested in modelling nondeterminism or not. It supports the verification of both CTL and LTL properties (when dealing with nonprobabilistic systems). This allowed us to express the formula $\exists\Diamond(\Box_{[t,t+5]} temp > 9.9)$ to verify violations of the safety conditions, avoiding the implementation of the *_Safety_* automaton. However, using integer variables to represent state variables with a fixed precision requires the introduction of extra transitions (to deal with nondeterministic errors), which significantly complicates the PRISM model. In this respect, Uppaal appears to be more efficient than PRISM, as we have been able to concisely express the error occurring in integer state variables thanks to the *select()* construct, in which the user can fix the granularity adopted to approximate a dense interval. This discrete representation provides an *under-approximation* of the system behaviour; thus, a finer granularity translates into an exponential increase of the complexity of the system, with obvious consequences on the verification performance. Then, we tried to model our case study in Real-Time Maude, a completely different framework for real-time systems, based on *rewriting logic*. The language supports object-like inheritance features that are quite helpful to represent complex systems in a modular manner. We used communication channels to implement our attacks on the physical devices. Furthermore, we used rational variables for a more concise discrete representation of state variables. We have been able to verify LTL and T-CTL properties, although the verification process resulted to be quite slow due to a proliferation of rewriting rules when fixing a reasonable granularity to approximate dense intervals. As the verification logic is quite powerful, there is no need to implement an ad hoc process to check for safety. Finally, we also tried to model our case study

in the *safety model checker* prohver within the MODEST TOOLSET (see [38]). We specified our case study in the high-level language HMODEST, supporting: (i) differential inclusion to model linear CPSs with constant bounded derivatives; (ii) linear formulae to express nondeterministic assignments within a dense interval; (iii) a compositional programming style inherited from process algebra; (iv) shared actions to synchronise parallel components. However, we faced the same performance limitations encountered in UPPAAL. Thus, we decided to move to statistical model checking.

Finally, this article extends the preliminary conference version [40] in the following aspects: (i) the calculus has been slightly redesigned by distinguishing physical state and physical environment, adding specifying constructs to sniff, drop and forge packets, and removing, for simplicity, protected physical devices; (ii) the two trace semantics have been proven to be compositional, i.e., preserved by properly defined contexts; (iii) both our running example *Sys* and the attacks proposed in Examples 1, 2, 3 and 4 have been implemented and verified in UPPAAL SMC.

6.3 Future work

While much is still to be done, we believe that our paper provides a stepping stone for the development of formal and automated tools to analyse the security of CPSs. We will consider applying, possibly after proper enhancements, existing tools and frameworks for automated security protocol analysis, resorting to the development of a dedicated tool if existing ones prove not up to the task. We will also consider further security properties and concrete examples of CPSs, as well as other kinds of physics-based attacks,such as *delays in the communication* of measurements and/or commands, and *periodic attacks*, i.e., attacks that operate in a periodic fashion inducing periodic physical effects on the targeted system that may be easily confused by engineers with system malfunctions. This will allow us to refine the classes of attacks we have given here (e.g., by formalising a type system amenable to static analysis), and provide a formal definition of when a CPS is more secure than another so as to be able to design, by progressive refinement, secure variants of a vulnerable CPSs.

We also aim to extend the behavioural theory of `CCPSA` by developing suitable *probabilistic metrics* to take into consideration the probability of a specific trace to actually occur. We have already done some progress in this direction for a variant of `CCPSA` with no security features in it, by defining ad hoc compositional *bisimulation metrics* [42]. In this manner, we believe that our notion of impact might be refined by taking into account quantitative aspects of an attack such as the probability of being successful when targeting a specific CPS. A first attempt on a (much) simpler IoT setting can be found in [41].

Finally, with respect to automatic approximations of the impact, while we have not yet fully investigated the problem, we believe that we can transform it into a "minimum problem". For instance, if the environment uses linear functions, then, by adapting techniques developed for linear hybrid automata (see, e.g., [3]), the set of all traces with length at most n (for a fixed n) can be characterised by a system of first degree inequalities, so the measure of the impact could be translated into a linear programming problem.

Acknowledgements

We thank the anonymous reviewers for their insightful and careful reviews. Massimo Merro and Andrei Munteanu have been partially supported by the project "Dipartimenti di Eccellenza 2018–2022" funded by the Italian Ministry of Education, Universities and Research (MIUR).

References

[1] R. Akella, H. Tang, and B. M. McMillin. Analysis of information flow security in cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 3(3–4):157–173, 2010.

[2] R. Alguliyev, Y. Imamverdiyev, and L. Sukhostat. Cyber-physical systems and their security issues. *Computers in Industry*, 100:212–223, 2018.

[3] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[4] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan. Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In *Lectures on Runtime Verification — Introductory and Advanced Topics*, LNCS 10457, pages 135–175. Springer, 2018.

[5] G. Behrmann, A. David, and K. G. Larsen. A Tutorial on Uppaal. In *Formal Methods for the Design of Real-Time Systems. SFM-RT 2004*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.

[6] G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *Quantitative Evaluation of Systems*, pages 125–126. IEEE Computer Society, 2006.

[7] J. A. Bergstra and C. A. Middelburg. Process Algebra for Hybrid Systems. *Theoretical Computer Science*, 335(2–3):215–280, 2005.

[8] C. Bodei, S. Chessa, and L. Galletta. Measuring security in iot communications. *Theoretical Computer Science*, pages 100–124, 2019.

[9] C. Bodei, P. Degano, G.-L. Ferrari, and L. Galletta. Tracing where IoT data are collected and aggregated. *Logical Methods in Computer Science*, 13(3:5):1–38, 2019.

[10] B. Bohrer and A. Platzer. A Hybrid, Dynamic Logic for Hybrid-Dynamic Information Flow. In *ACM/IEEE Symposium on Logic in Computer Science*, pages 115–124. ACM, 2018.

[11] M. Burmester, E. Magkos, and V. Chrissikopoulos. Modeling security in cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 5(3–4):118–126, 2012.

[12] H. Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[13] L. F. Cómbita, J. Giraldo, A. A. Cárdenas, and N. Quijano. Response and reconfiguration of cyber-physical control systems: A survey. In *Colombian Conference on Automatic Control*, pages 1–6. IEEE, 2015.

[14] P. J. L. Cuijpers and M. A. Reniers. Hybrid process algebra. *The Journal of Logic and Algebraic Programming*, 62(2):191–245, 2005.

[15] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen. Uppaal SMC Tutorial. *International Journal on Software Tools for Technology Transfer*, 17(4):397–415, 2015.

[16] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on information theory*, (2):198–208, 1983.

[17] R. Focardi and F. Martinelli. A Uniform Approach for the Definition of Security Properties. In *Formal Methods*, volume 1708 of *Lecture Notes in Computer Science*, pages 794–813. Springer, 1999.

[18] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. *International Journal on Software Tools for Technology Transfer*, 10(3):263–279, 2008.

[19] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable Verification of Hybrid Systems. In *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2011.

[20] V. Galpin, L. Bortolussi, and J. Hillston. HYPE: Hybrid modelling by composition of flows. *Formal Aspects of Computing*, 25(4):503–541, 2013.

[21] B. Genge, I. Kiss, and P. Haller. A system dynamics approach for assessing the impact of cyber attacks on critical infrastructures. *International Journal of Critical Infrastructure Protection*, 10:3–17, 2015.

[22] J. Giraldo, E. Sarkar, A. A. Cárdenas, M. Maniatakos, and M. Kantarcioglu. Security and Privacy in Cyber-Physical Systems: A Survey of Surveys. *IEEE Design & Test*, 34(4):7–17, 2017.

[23] J. Giraldo, D. I. Urbina, A. A. Cárdenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell. A Survey of Physics-Based Attack Detection in Cyber-Physical Systems. *ACM Computing Surveys (CSUR)*, 51(4):76:1–76:36, 2018.

[24] D. Gollmann, P. Gurikov, A. Isakov, M. Krotofil, J. Larsen, and A. Winnicki. Cyber-Physical Systems Security: Experimental Analysis of a Vinyl Acetate Monomer Plant. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pages 1–12. ACM, 2015.

[25] D. Gollmann and M. Krotofil. Cyber-Physical Systems Security. In *The New Codebreakers – Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, volume 9100 of *Lecture Notes in Computer Science*, pages 195–204. Springer, 2016.

[26] A. Hartmanns and H. Hermanns. The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *Lecture Notes in Computer Science*, pages 593–598. Springer, 2014.

[27] M. Hennessy and T. Regan. A Process Algebra for Timed Systems. *Information and Computation*, 117(2):221–239, 1995.

[28] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A Model Checker for Hybrid Systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.

[29] K. Huang, C. Zhou, Y.-C. Tian, S. Yang, and Y. Qin. Assessing the Physical Impact of Cyberattacks on Industrial Cyber-Physical Systems. *IEEE Transactions on Industrial Electronics*, 65(10):8153–8162, 2018.

[30] Y.-L. Huang, A. A. Cárdenas, S. Amin, Z.-S. Lin, H.-Y. Tsai, and S. Sastry. Understanding the physical and economic consequences of attacks on control systems. *International Journal of Critical Infrastructure Protection*, 2(3):73–83, 2009.

[31] ICS-CERT. Cyber-Attack Against Ukrainian Critical Infrastructure. https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01, 2015.

[32] M. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. Talcott. Discrete vs. Dense Times in the Analysis of Cyber-Physical Security Protocols. In *Principles of Security and Trust*, volume 9036 of *Lecture Notes in Computer Science*, pages 259–279. Springer, 2015.

[33] M. Krotofil and A. A. Cárdenas. Resilience of Process Control Systems to Cyber-Physical Attacks. In *NordSec 2013: Secure IT Systems*, volume 8208 of *Lecture Notes in Computer Science*, pages 166–182. Springer, 2013.

[34] M. Krotofil, A. A. Cárdenas, J. Larsen, and D. Gollmann. Vulnerabilities of cyber-physical systems to stale data – Determining the optimal time to launch attacks. *International Journal of Critical Infrastructure Protection*, 7(4):213–232, 2014.

[35] D. Kushner. The real story of stuxnet. *IEEE Spectrum*, 50(3):48–53, 2013.

[36] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[37] R. Lanotte and M. Merro. A Calculus of Cyber-Physical Systems. In *Language and Automata Theory and Applications*, volume 10168 of *Lecture Note in Computer Science*, pages 115–127. Springer, 2017.

[38] R. Lanotte, M. Merro, and A. Munteanu. A Modest Security Analysis of Cyber-Physical Systems: A Case Study. In *Formal Techniques for Distributed Objects, Components, and Systems*, volume 10854 of *Lecture Notes in Computer Science*, pages 58–78. Springer, 2018.

[39] R. Lanotte, M. Merro, A. Munteanu, and L. Viganò. A Formal Approach to Physics-Based Attacks in Cyber-Physical Systems (Extended Version). *CoRR*, abs/1902.04572, 2019.

[40] R. Lanotte, M. Merro, R. Muradore, and L. Viganò. A Formal Approach to Cyber-Physical Attacks. In *Computer Security Foundations Symposium*, pages 436–450. IEEE Computer Society, 2017.

[41] R. Lanotte, M. Merro, and S. Tini. Towards a Formal Notion of Impact Metric for Cyber-Physical Attacks. In *Integrated Formal Methods*, volume 11023 of *Lecture Notes in Computer Science*, pages 296–315. Springer, 2018.

[42] R. Lanotte, M. Merro, and S. Tini. A Probabilistic Calculus of Cyber-Physical Systems. *Information and Computation*, 2020.

[43] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux. Game theory meets network security and privacy. *ACM Computer Surveys*, 45(3):25, 2013.

[44] J. Milošević, D. Umsonst, H. Sandberg, and K. H. Johansson. Quantifying the Impact of Cyber-Attack Strategies for Control Systems Equipped With an Anomaly Detector. In *European Control Conference (ECC)*, pages 331–337. IEEE, 2018.

[45] A. F. Murillo Piedrahita, V. Gaur, J. Giraldo, A. A. Cárdenas, and S. J. Rueda. Virtual incident response functions in control systems. *Computer Networks*, 135:147–159, 2018.

[46] V. Nigam, C. Talcott, and A. A. Urquiza. Towards the Automated Verification of Cyber-Physical Security Protocols: Bounding the Number of Timed Intruders. In *Computer Security - ESORICS 2016*, volume 9879 of *Lecture Notes in Computer Science*, pages 450–470. Springer, 2016.

[47] P. C. Ölveczky and J. Meseguer. Semantics and pragmatics of Real-Time Maude. *Higher-Order and Symbolic Computation*, 20(1–2):161–196, 2007.

[48] A. Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, 2018.

[49] J.-D. Quesel, S. Mitsch, S. M. Loos, N. Aréchiga, and A. Platzer. How to model and prove hybrid systems with KeYmaera: a tutorial on safety. *International Journal on Software Tools for Technology Transfer*, 18(1):67–91, 2016.

[50] M. Rocchetto and N. O. Tippenhauer. CPDY: Extending the Dolev-Yao Attacker with Physical-Layer Interactions. In *Formal Methods and Software Engineering*, volume 10009 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2016.

[51] M. Rocchetto and N. O. Tippenhauer. On Attacker Models and Profiles for Cyber-Physical Systems. In *Computer Security - ESORICS 2016*, volume 9879 of *Lecture Notes in Computer Science*, pages 427–449. Springer, 2016.

[52] W. C. Rounds and H. Song. The ϕ-calculus: A Language for Distributed Control of Reconfigurable Embedded Systems. In *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 435–449. Springer, 2003.

[53] J. Slay and M. Miller. Lessons Learned from the Maroochy Water Breach. In *Critical Infrastructure Protection*, IFIP 253, pages 73–82. Springer, 2007.

[54] Swedish Civil Contigencies Agency. Guide to increased security in industrial information and control systems. 2014.

[55] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson. A secure control framework for resource-limited adversaries. *Automatica*, 51:135–148, 2015.

[56] U.S. Chemical Safety and Hazard Investigation Board, T2 Laboratories Inc. Reactive Chemical Explosion: Final Investigation Report. Report No. 2008-3-I-FL, 2009.

[57] D. A. van Beek, K. L. Man, M. A. Reniers, J. E. Rooda, and R. R. Schiffelers. Syntax and consistent equation semantics of hybrid chi. *The Journal of Logic and Algebraic Programming*, 68(1-2):129–210, 2006.

[58] P. Vasilikos, F. Nielson, and H. Riis Nielson. Secure Information Release in Timed Automata. In *Principles of Security and Trust*, volume 10804 of *Lecture Notes in Computer Science*, pages 28–52. Springer, 2018.

[59] R. Vigo. The Cyber-Physical Attacker. In *Computer Safety, Reliability, and Security*, volume 7613 of *Lecture Notes in Computer Science*, pages 347–356. Springer, 2012.

[60] R. Vigo, F. Nielson, and H. Riis Nielson. Broadcast, Denial-of-Service, and Secure Communication. In *Integrated Formal Methods*, volume 7940 of *Lecture Notes in Computer Science*, pages 412–427. Springer, 2013.

[61] J. Wang and H. Yu. Analysis of the Composition of Non-Deducibility in Cyber-Physical Systems. *Applied Mathematics & Information Sciences*, 8:3137–3143, 2014.

[62] Y. Zacchia Lun, A. D'Innocenzo, I. Malavolta, and M. D. Di Benedetto. Cyber-Physical Systems Security: a Systematic Mapping Study. *CoRR*, abs/1605.09641, 2016.

[63] Y. Zacchia Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto. State of the art of cyber-physical systems security: An automatic control perspective. *Journal of Systems and Software*, 149:174–216, 2019.

[64] Q. Zhu and T. Basar. Game-Theoretic Methods for Robustness, Security, and Resilience of Cyberphysical Control Systems: Games-in-Games Principle for Optimal Cross-Layer Resilient Control Systems. *IEEE Control Systems Magazine*, 35(1):46–65, 2015.

A Proofs

As already stated in Remark 2, our trace preorder $\sqsubseteq$ is deadlock-sensitive. Formally,

Lemma 1. *Let M and N be two CPSs in* `CCPSA` *such that $M \sqsubseteq N$. Then, M satisfies its system invariant if and only if N satisfies its system invariant.*

Proof. This is because CPSs that don't satisfy their invariant can only fire deadlock actions. □

Proof of Theorem 1. We prove the three statements separately.

1. Let us prove that $M \uplus O \xrightarrow{t} M' \uplus O'$ entails $N \uplus O \stackrel{\hat{t}}{\Longrightarrow} N' \uplus O'$. The proof is by induction on the length of the trace $M \uplus O \xrightarrow{t} M' \uplus O'$.
   As $M \sqsubseteq N$, by an application of Lemma 1 it follows that either both M and N satisfy their respective invariants or they both don't. In the latter case, the result would be easy to prove as the systems can

only fire deadlock actions. Similarly, if the system invariant of O is not satisfied, then $M \uplus O$ and $N \uplus O$ can perform only deadlock actions and again the result would follow easily. Thus, let us suppose that the system invariants of M, N and O are satisfied.
Base case. We suppose $M = E_1; S_1 \bowtie P_1$, $N = E_2; S_2 \bowtie P_2$, and $O = E_3; S_3 \bowtie P_3$. We proceed by case analysis on why $M \uplus O \xrightarrow{\alpha} M' \uplus O'$, for some action α.

- $\alpha = \overline{c}v$. Suppose $M \uplus O \xrightarrow{\overline{c}v} M' \uplus O'$ is derived by an application of rule (Out). We have two possible cases:
  - either $P_1 \parallel P_3 \xrightarrow{\overline{c}v} P_1 \parallel P_3'$, because $P_3 \xrightarrow{\overline{c}v} P_3'$, for some P_3', $O' = S_3 \bowtie P_3'$, and $M' = M$,
  - or $P_1 \parallel P_3 \xrightarrow{\overline{c}v} P_1' \parallel P_3$, because $P_1 \xrightarrow{\overline{c}v} P_1'$, for some P_1', and $M = S_1 \bowtie P_1'$ and $O' = O$.

  In the first case, by an application of rule (Par) we derive $P_2 \parallel P_3 \xrightarrow{\overline{c}v} P_2 \parallel P_3'$. Since both system invariants of N and O are satisfied, we can derive the required trace $N \uplus O \xrightarrow{\overline{c}v} N \uplus O'$ by an application of rule (Out). In the second case, since $P_1 \xrightarrow{\overline{c}v} P_1'$ and the invariant of M is satisfied, by an application of rule (Out) we can derive $M \xrightarrow{\overline{c}v} M'$. As $M \sqsubseteq N$, there exists a trace $N \xRightarrow{\widehat{\overline{c}v}} N'$, for some system N'. Thus, by several applications of rule (Par) we can easily derive $N \uplus O \xRightarrow{\widehat{\overline{c}v}} N' \uplus O = N' \uplus O'$, as required.
- $\alpha = cv$. Suppose $M \uplus O \xrightarrow{cv} M' \uplus O'$ is derived by an application of rule (Inp). This case is similar to the previous one.
- $\alpha = \tau$. Suppose $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule (SensRead). We have two possible cases:
  - either $P_1 \parallel P_3 \xrightarrow{s?v} P_1 \parallel P_3'$ because $P_3 \xrightarrow{s?v} P_3'$, for some P_3', $P_1 \parallel P_3 \xrightarrow{\lightning s!v \not\rightarrow}$ (and hence $P_3 \xrightarrow{\lightning s!v \not\rightarrow}$), $M' = M$ and $O' = S_3 \bowtie P_3'$,
  - or $P_1 \parallel P_3 \xrightarrow{s?v} P_1' \parallel P_3$ because $P_1 \xrightarrow{s?v} P_1'$, for some P_1', $P_1 \parallel P_3 \xrightarrow{\lightning s!v \not\rightarrow}$ (and hence $P_1 \xrightarrow{\lightning s!v \not\rightarrow}$) and $M' = S_1 \bowtie P_1'$ and $O' = O$.

  In the first case, by an application of rule (Par) we derive $P_2 \parallel P_3 \xrightarrow{s?v} P_2 \parallel P_3'$. Moreover from $P_3 \xrightarrow{\lightning s!v \not\rightarrow}$ and since the sets of sensors are always disjoint, we can derive $P_2 \parallel P_3 \xrightarrow{\lightning s!v \not\rightarrow}$. Since both invariants of N and O are satisfied, we can derive $N \uplus O \xrightarrow{\tau} N \uplus O'$ by an application of rule (SensRead), as required. In the second case, since $P_1 \xrightarrow{s?v} P_1'$ and the invariant of M is satisfied, by an application of rule (SensRead) we can derive $M \xrightarrow{\tau} M'$ with $M' = S_1 \bowtie P_1'$. As $M \sqsubseteq N$, there exists a derivation $N \xRightarrow{\hat{\tau}} N'$, for some N'. Thus, we can derive the required trace $N \uplus O \xRightarrow{\hat{\tau}} N' \uplus O$ by an application of rule (Par).
- $\alpha = \tau$. Suppose that $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule (↯SensSniff↯). This case is similar to the previous one.
- $\alpha = \tau$. Suppose that $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule (ActWrite). This case is similar to the case (SensRead).
- $\alpha = \tau$. Suppose that $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule (↯AcIntegr↯). This case is similar to the case (↯SensSniff↯).
- $\alpha = \tau$. Suppose that $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule (Tau). We have four possible cases:
  - $P_1 \parallel P_3 \xrightarrow{\tau} P_1' \parallel P_3'$ by an application of rule (Com). We have two sub-cases: either $P_1 \xrightarrow{\overline{c}v} P_1'$ and $P_3 \xrightarrow{cv} P_3'$, or $P_1 \xrightarrow{cv} P_1'$ and $P_3 \xrightarrow{\overline{c}v} P_3'$, for some P_1' and P_3'. We prove the first case, the second one is similar. As the invariant of M is satisfied, by an application

of rule (Out) we can derive $M \xrightarrow{\overline{c}v} M'$. As $M \sqsubseteq N$, there exists a trace $N \stackrel{\hat{\tau}}{\Longrightarrow} \xrightarrow{\overline{c}v} \stackrel{\hat{\tau}}{\Longrightarrow} N'$, for some $N' = E_2; S_2' \bowtie P_2'$. As $P_3 \xrightarrow{cv} P_3'$, by several applications of rule (Par) and one of rule (Com) we derive $N \uplus O \stackrel{\widehat{\overline{c}v}}{\Longrightarrow} N' \uplus O'$, as required.
  - $P_1 \parallel P_3 \xrightarrow{\tau} P_1 \parallel P_3'$ or $P_1 \parallel P_3 \xrightarrow{\tau} P_1' \parallel P_3$ by an application of (Par). This case is easy.
  - $P_1 \parallel P_3 \xrightarrow{\tau} P_1' \parallel P_3'$ by an application of either rule ($\lightning$ActDrop$\lightning$) or rule ($\lightning$SensIntegr$\lightning$). This case does not apply as the sets of actuators of M and O are disjoint.
  - $P_1 \parallel P_3 \xrightarrow{\tau} P_1' \parallel P_3'$ by the application of on rule among (Res), (Rec), (Then) and (Else). This case does not apply to parallel processes.

- $\alpha = \mathsf{deadlock}$. Suppose that $M \uplus O \xrightarrow{\mathsf{deadlock}} M' \uplus O'$ is derived by an application of rule (Deadlock). This case is not admissible as the invariants of M, N and O are satisfied.

- $\alpha = \mathsf{tick}$. Suppose that $M \uplus O \xrightarrow{\mathsf{tick}} M' \uplus O'$ is derived by an application of rule (Time). This implies $P_1 \parallel P_3 \xrightarrow{\mathsf{tick}} P_1' \parallel P_3'$, for some P_1' and P_3', $M' = E_1; S_1' \bowtie P_1'$ and $O = E_3; S_3' \bowtie P_3'$, with $S_1' \in \mathrm{next}(E_1; S_1)$ and $S_3' \in \mathrm{next}(E_3; S_3)$. As $P_1 \parallel P_3 \xrightarrow{\mathsf{tick}} P_1' \parallel P_3'$ can only be derived by an application of rule (TimePar), it follows that $P_1 \xrightarrow{\mathsf{tick}} P_1'$ and $P_3 \xrightarrow{\mathsf{tick}} P_3'$. Since the invariant of M is satisfied, by an application of rule (Time) we can derive $M \xrightarrow{\mathsf{tick}} M'$ with $M' = E_1; S_1' \bowtie P_1'$. As $M \sqsubseteq N$, there exists a derivation $N \stackrel{\hat{\tau}}{\Longrightarrow} N'' \xrightarrow{\mathsf{tick}} N''' \stackrel{\hat{\tau}}{\Longrightarrow} N'$, for some $N' = E_2; S_2' \bowtie P_2'$, $N'' = E_2; S_2'' \bowtie P_2''$, $N''' = E_2; S_2''' \bowtie P_2'''$, with $S_2''' \in \mathrm{next}(E_2; S_2'')$. By several applications of rule (Par) we can derive that $N \uplus O \stackrel{\hat{\tau}}{\Longrightarrow} N'' \uplus O$ and $N''' \uplus O' \stackrel{\hat{\tau}}{\Longrightarrow} N' \uplus O'$. In order to conclude the proof, it is sufficient to prove $N'' \uplus O \xrightarrow{\mathsf{tick}} N''' \uplus O'$. By the definition of rule (Time), from $N'' \xrightarrow{\mathsf{tick}} N'''$ it follows that $P_2'' \xrightarrow{\mathsf{tick}} P_2'''$. As $P_3 \xrightarrow{\mathsf{tick}} P_3'$, by an application of rule (TimePar) it follows that $P_2'' \parallel P_3 \xrightarrow{\mathsf{tick}} P_2''' \parallel P_3'$. Since $S_2''' \in \mathrm{next}(E_2; S_2'')$ and $S_3' \in \mathrm{next}(E_3; S_3)$ we can derive that $S_2''' \uplus S_3' \in \mathrm{next}(E_2; S_2'') \cup \mathrm{next}(E_3; S_3)$. By an application of rule (Time) we have $N'' \uplus O \xrightarrow{\mathsf{tick}} N''' \uplus O'$ and hence $N \uplus O \stackrel{\widehat{\mathsf{tick}}}{\Longrightarrow} N' \uplus O'$, as required.

- $\alpha = \mathsf{unsafe}$. Suppose that $M \uplus O \xrightarrow{\mathsf{unsafe}} M' \uplus O'$ is derived by an application of rule (Safety). This is similar to the case $\alpha = \overline{c}v$ by considering the fact that $\xi_{\mathrm{x}} \notin \mathit{safe}$ implies that $\xi_{\mathrm{x}} \cup \xi_{\mathrm{x}}{}' \notin \mathit{safe} \cup \mathit{safe}'$, for any $\xi_{\mathrm{x}}{}'$ and any safe'.

Inductive case. We have to prove that $M \uplus O = M_0 \uplus O_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} M_n \uplus O_n$ implies $N \uplus O = N_0 \uplus O_0 \stackrel{\widehat{\alpha_1}}{\Longrightarrow} \cdots \stackrel{\widehat{\alpha_n}}{\Longrightarrow} N_n \uplus O_n$. We can use the inductive hypothesis to easily deal with the first $n-1$ actions and resort to the base case to handle the n^{th} action.

2. We have to prove that $M \sqsubseteq N$ implies $M \parallel P \sqsubseteq N \parallel P$, for any pure-logical process P. This is a special case of (1) as $M \parallel P = M \uplus (\emptyset; \emptyset \bowtie P)$ and $N \parallel P = N \uplus (\emptyset; \emptyset \bowtie P)$, where $\emptyset; \emptyset \bowtie P$ is a CPS with no physical process in it, only logics.
3. We have to prove that $M \sqsubseteq N$ implies $M \backslash c \sqsubseteq N \backslash c$, for any channel c. For any derivation $M \backslash c \xrightarrow{t} M' \backslash c$ we can easily derive that $M \xrightarrow{t} M'$ with c not occurring in t. Since $M \sqsubseteq N$, it follows that $N \stackrel{\hat{t}}{\Longrightarrow} N'$, for some N'. Since c does not appear in t, we can easily derive that $N \backslash c \stackrel{\hat{t}}{\Longrightarrow} N' \backslash c$, as required.

□

In order to prove Theorem 2 we adapt to CCPSA two standard lemmata used in process calculi theory to compose and decompose the actions performed by a compound system.

Lemma 2 (Decomposing system actions)**.** *Let M and N be two CPSs in CCPSA. Then,*

- *if $M \uplus N \xrightarrow{\mathsf{tick}} M' \uplus N'$, for some M' and N', then $M \xrightarrow{\mathsf{tick}} M'$ and $N \xrightarrow{\mathsf{tick}} N'$;*
- *if $M \uplus N \xrightarrow{\mathsf{deadlock}} M \uplus N$, then $M \xrightarrow{\mathsf{deadlock}} M$ or $N \xrightarrow{\mathsf{deadlock}} N$;*

- *if* $M \uplus N \xrightarrow{\tau} M' \uplus N'$, *for some* M' *and* N', *due to a channel synchronisation between* M *and* N, *then either* $M \xrightarrow{\overline{c}v} M'$ *and* $N \xrightarrow{cv} N'$, *or* $M \xrightarrow{cv} M'$ *and* $N \xrightarrow{\overline{c}v} N'$, *for some channel* c;
- *if* $M \uplus N \xrightarrow{\alpha} M' \uplus N'$, *for some* M' *and* N', $\alpha \neq \mathsf{tick}$, *not due to a channel synchronisation between* M *and* N, *then either* $M \xrightarrow{\alpha} M$ *and* $N = N'$, *or* $N \xrightarrow{\alpha} N$ *and* $M = M'$.

Lemma 3 (Composing system actions)**.** *Let* M *and* N *be two CPSs of* `CCPSA`. *Then,*

- *If* $M \xrightarrow{\mathsf{tick}} M'$ *and* $N \xrightarrow{\mathsf{tick}} N'$, *for some* M' *and* N', *then* $M \uplus N \xrightarrow{\mathsf{tick}} M' \uplus N'$;
- *If* $N \xrightarrow{\mathsf{deadlock}} \not\rightarrow$ *and* $M \xrightarrow{\alpha} M'$, *for some* M' *and* $\alpha \neq \mathsf{tick}$, *then* $M \uplus N \xrightarrow{\alpha} M' \uplus N$ *and* $N \uplus M \xrightarrow{\alpha} N \uplus M'$.

Proof of Theorem 2. Here, we prove case (1) of the theorem. The proofs of cases (2) and (3) are similar to the corresponding ones of Theorem 1.

We prove that $M \sqsubseteq_{m..n} N$ implies that there are $m', n' \in \mathbb{N}^+ \cup \infty$, with $m'..n' \subseteq m..n$ such that $M \uplus O \sqsubseteq_{m'..n'} N \uplus O$. We prove separately that $m' \geq m$ and $n' \leq n$.

- $m' \geq m$. We recall that $m, m' \in \mathbb{N}^+$. If $m = 1$, then we trivially have $m' \geq 1 = m$. Otherwise, since m is the minimum integer for which there is a trace t, with $\#\mathsf{tick}(t) = m - 1$, such that $M \xrightarrow{t}$ and $N \stackrel{\hat{t}}{\not\Longrightarrow}$, then for any trace t, with $\#\mathsf{tick}(t) < m - 1$ and such that $M \xrightarrow{t}$, it holds that $N \stackrel{\hat{t}}{\Longrightarrow}$. As done in the proof of case (1) of Theorem 1, we can derive that for any trace t, with $\#\mathsf{tick}(t) < m - 1$ and such that $M \uplus O \xrightarrow{t}$ it holds that $N \uplus O \stackrel{\hat{t}}{\Longrightarrow}$. This implies the required condition, $m' \geq m$.

- $n' \leq n$. We recall that n is the infimum element of $\mathbb{N}^+ \cup \{\infty\}$, $n \geq m$, such that whenever $M \xrightarrow{t_1} M'$, with $\#\mathsf{tick}(t_1) = n - 1$, there is t_2, with $\#\mathsf{tick}(t_1) = \#\mathsf{tick}(t_2)$, such that $N \xrightarrow{t_2} N'$, for some N', and $M' \sqsubseteq N'$. Now, if $M \uplus O \xrightarrow{t} M' \uplus O'$, with $\#\mathsf{tick}(t) = n - 1$, by Lemma 2 we can split the trace t by extracting the actions performed by M and those performed by O. Thus, there exist two traces $M \xrightarrow{t_1} M'$ and $O \xrightarrow{t_3} O'$, with $\#\mathsf{tick}(t_1) = \#\mathsf{tick}(t_3) = n - 1$ whose combination has generated the trace $M \uplus O \xrightarrow{t} M' \uplus O'$. As $M \sqsubseteq_{m..n} N$, from $M \xrightarrow{t_1} M'$ we know that there is a trace t_2, with $\#\mathsf{tick}(t_1) = \#\mathsf{tick}(t_2)$, such that $N \xrightarrow{t_2} N'$, for some N', and $M' \sqsubseteq N'$. Since $N \xrightarrow{t_2} N'$ and $O \xrightarrow{t_3} O'$, by an application of Lemma 3 we can build a trace $N \uplus O \xrightarrow{t'} N' \uplus O'$, for some t' such that $\#\mathsf{tick}(t) = \#\mathsf{tick}(t') = n - 1$. As $M' \sqsubseteq N'$, by Theorem 1 we can derive that $M' \uplus O' \sqsubseteq N' \uplus O'$. This implies that $n' \leq n$.

□

In order to prove Theorem 3, we introduce the following lemma.

Lemma 4. *Let* M *be an honest and sound CPS,* C *an arbitrary class of attacks, and* A *an attack of a class* $C' \preceq C$. *Whenever* $M \parallel A \xrightarrow{t} M' \parallel A'$, *then* $M \parallel Top(C) \stackrel{\hat{t}}{\Longrightarrow} M' \parallel \prod_{\iota \in \mathcal{I}} Att(\iota, \#\mathsf{tick}(t)+1, C(\iota))$.

Proof. Let us define $Top^h(C)$ as the attack process $\prod_{\iota \in \mathcal{I}} Att(\iota, h, C(\iota))$. Then, $Top^1(C) = Top(C)$. The proof is by mathematical induction on the length k of the trace t.
Base case. $k = 1$. This means $t = \alpha$, for some action α. We proceed by case analysis on α.

- $\alpha = \overline{c}v$. As the attacker A does not use communication channels, from $M \parallel A \xrightarrow{\overline{c}v} M' \parallel A'$ we can derive that $A = A'$ and $M \xrightarrow{\overline{c}v} M'$. Thus, by applications of rules (Par) and (Out) we derive $M \parallel Top(C) \xrightarrow{\overline{c}v} M' \parallel Top^1(C) = M' \parallel Top(C)$.
- $\alpha = cv$. This case is similar to the previous one.
- $\alpha = \tau$. There are five sub-cases.

– Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule (SensRead). Since the attacker A performs only malicious actions, from $M \parallel A \xrightarrow{\tau} M' \parallel A'$ we can derive that $A = A'$ and $P \xrightarrow{s?v} P'$ for some process P and P' such that $M = E; S \bowtie P$ and $M' = E; S \bowtie P'$. By considering $rnd(\{\mathsf{true}, \mathsf{false}\}) = \mathsf{false}$ for any process $Att(\iota, 1, C(\iota))$, we have that $Top(C)$ can only perform a tick action, and $Top(C) \xrightarrow{\lightning s!v}$. Hence, by an application of rules (Par) and (SensRead) we derive $M \parallel Top(C) \xrightarrow{\tau} M' \parallel Top^1(C) = M' \parallel Top(C)$.

– Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule (ActWrite). This case is similar to the previous one.

– Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule ($\lightning$SensSniff$\lightning$). Since M is sound it follows that $M = M'$ and $A \xrightarrow{\lightning s?v} A'$. This entails $1 \in C'(\lightning s?) \subseteq C(\lightning s?)$. By assuming $rnd(\{\mathsf{true}, \mathsf{false}\}) = \mathsf{true}$ for the process $Att(\lightning s?, 1, C(\lightning s?))$, it follows that $Top(C) \xrightarrow{\lightning s?v} Top^1(C) = Top(C)$. Hence, by applying the rules (Par) and ($\lightning$SensRead$\lightning$) we derive $M \parallel Top(C) \xrightarrow{\tau} M' \parallel Top^1(C) = M' \parallel Top(C)$.

– Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule ($\lightning$ActIntgr$\lightning$). Since M is sound it follows that $M = M'$ and $A \xrightarrow{\lightning a!v} A'$. As a consequence, $1 \in C'(\lightning a!) \subseteq C(\lightning a!)$. By assuming $rnd(\{\mathsf{true}, \mathsf{false}\})$=$\mathsf{true}$ and $rnd(\mathbb{R}) = v$ for the process $Att(\lightning a!, 1, C(\lightning a!))$, it follows that $Top(C) \xrightarrow{\lightning a!v} Top^1(C) = Top(C)$. Thus, by applying the rules (Par) and ($\lightning$ActIntegr$\lightning$) we derive $M \parallel Top(C) \xrightarrow{\tau} M' \parallel Top^1(C) = M' \parallel Top(C)$.

– Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule (Tau). Let $M = E; S \bowtie P$ and $M' = E'; S \bowtie P'$. First, we consider the case when $P \parallel A \xrightarrow{\tau} P' \parallel A'$ is derived by an application of either rule ($\lightning$SensIntegr$\lightning$) or rule ($\lightning$ActDrop$\lightning$). Since M is sound and A can perform only malicious actions, we have that: (i) either $P \xrightarrow{s?v} P'$ and $A \xrightarrow{\lightning s!v} A'$ (ii) or $P \xrightarrow{a!v} P'$ and $A \xrightarrow{\lightning a?v} A'$. We focus on the first case as the second one is similar.
Since $A \xrightarrow{\lightning s!v} A'$, we derive $1 \in C'(\lightning s!) \subseteq C(\lightning s!)$, and $Top(C) \xrightarrow{\lightning s!v} Top^1(C) = Top(C)$, by assuming $rnd(\{\mathsf{true}, \mathsf{false}\}) = \mathsf{true}$ and $rnd(\mathbb{R}) = v$ for the process $Att(\lightning s!, 1, C(\lightning s!))$. Thus, by applying the rules ($\lightning$SensIntegr$\lightning$) and (Tau) we derive $M \parallel Top(C) \xrightarrow{\tau} M' \parallel Top^1(C) = M' \parallel Top(C)$, as required.
To conclude the proof we observe that if $P \parallel A \xrightarrow{\tau} P' \parallel A'$ is derived by an application of a rule different from ($\lightning$SensIntegr$\lightning$) and ($\lightning$ActDrop$\lightning$), then by inspection of Table 1 and by definition of attacker, it follows that A can't perform a τ-action since A does not use channel communication and performs only malicious actions. Thus, the only possibility is that the τ-action is performed by P in isolation. As a consequence, by applying the rules (Par) and (Tau), we derive $M \parallel Top(C) \xrightarrow{\tau} M' \parallel Top^1(C) = M' \parallel Top(C)$.

- $\alpha = \mathsf{tick}$. In this case the transition $M \parallel A \xrightarrow{\mathsf{tick}} M' \parallel A'$ is derived by an application of rule (Time) because $M \xrightarrow{\mathsf{tick}} M'$ and $A \xrightarrow{\mathsf{tick}} A'$. Hence, it suffices to prove that $Top(C) \xrightarrow{\mathsf{tick}} Top^2(C)$. We consider two cases: $1 \in C(\iota)$ and $1 \notin C(\iota)$. If $1 \in C(\iota)$, then the transition $Att(\iota, 1, C(\iota)) \xrightarrow{\mathsf{tick}} Att(\iota, 2, C(\iota))$ can be derived by assuming $rnd(\{\mathsf{true}, \mathsf{false}\}) = \mathsf{false}$. Moreover, since $rnd(\{\mathsf{true}, \mathsf{false}\}) = \mathsf{false}$ the process $Att(\iota, 1, C(\iota))$ can only perform a tick action. If $1 \notin C(\iota)$, then the process $Att(\iota, 1, C(\iota))$ can only perform a tick action. As a consequence, $Att(\iota, 1, C(\iota)) \xrightarrow{\mathsf{tick}} Att(\iota, 2, C(\iota))$ and $Top(C) \xrightarrow{\mathsf{tick}} Top^2(C)$. By an application of rule (Time), we derive $M \parallel Top(C) \xrightarrow{\mathsf{tick}} M' \parallel Top^2(C)$.

- $\alpha = \mathsf{deadlock}$. This case is not admissible because $M \parallel A \xrightarrow{\mathsf{deadlock}} M' \parallel A'$ would entail $M \xrightarrow{\mathsf{deadlock}} M'$. However, M is sound and it can't deadlock.

- $\alpha = \mathsf{unsafe}$. Again, this case is not admissible because M is sound.

Inductive case ($k > 1$). We have to prove that $M \parallel A \xrightarrow{t} M' \parallel A'$ implies $M \parallel \mathit{Top}(C) \stackrel{\hat{t}}{\Longrightarrow} M' \parallel \mathit{Top}^{\#\mathsf{tick}(t)+1}(C)$. Since the length of t is greater than 1, it follows that $t = t'\alpha$, for some trace t' and some action α. Thus, there exist M'' and A'' such that $M \parallel A \xrightarrow{t'} M'' \parallel A'' \xrightarrow{\alpha} M' \parallel A'$. By inductive hypothesis, it follows that $M \parallel \mathit{Top}(C) \stackrel{\hat{t'}}{\Longrightarrow} M'' \parallel \mathit{Top}^{\#\mathsf{tick}(t')+1}(C)$. To conclude the proof, it is enough to show that $M'' \parallel A'' \xrightarrow{\alpha} M' \parallel A'$ implies $M'' \parallel \mathit{Top}^{\#\mathsf{tick}(t')+1}(C) \stackrel{\hat{\alpha}}{\Longrightarrow} M' \parallel \mathit{Top}^{\#\mathsf{tick}(t)+1}(C)$. The reasoning is similar to that followed in the base case, except for actions $\alpha = \mathsf{deadlock}$ and $\alpha = \mathsf{unsafe}$ that need to be treated separately. We prove the case $\alpha = \mathsf{deadlock}$ as the case $\alpha = \mathsf{unsafe}$ is similar.

Let $M = E; S \bowtie P$. The transition $M'' \parallel A \xrightarrow{\mathsf{deadlock}} M' \parallel A'$ must be derived by an application of rule (Deadlock). This implies that $M'' = M'$, $A'' = A'$ and the state function of M is not in the invariant set inv. Thus, by an application of rule (Deadlock) we derive

$$M'' \parallel \mathit{Top}^{\#\mathsf{tick}(t')+1}(C) \xrightarrow{\mathsf{deadlock}} M' \parallel \mathit{Top}^{\#\mathsf{tick}(t')+1}(C).$$

Since $\#\mathsf{tick}(t) + 1 = \#\mathsf{tick}(t') + \#\mathsf{tick}(\mathsf{deadlock}) + 1 = \#\mathsf{tick}(t') + 1$, it follows, as required, that

$$M'' \parallel \mathit{Top}^{\#\mathsf{tick}(t')+1}(C) \xrightarrow{\mathsf{deadlock}} M' \parallel \mathit{Top}^{\#\mathsf{tick}(t)+1}(C)\,.$$

□

Everything is finally in place to prove Theorem 3.

Proof of Theorem 3. We have to prove that either $M \parallel A \sqsubseteq M$ or $M \parallel A \sqsubseteq_{m_2..n_2} M$, for some m_2 and n_2 such that $m_2..n_2 \subseteq m_1..n_1$ ($m_2 = 1$ and $n_2 = \infty$ if the two systems are completely unrelated). The proof proceeds by contradiction. Suppose that $M \parallel A \not\sqsubseteq M$ and $M \parallel A \sqsubseteq_{m_2..n_2} M$, with $m_2..n_2 \not\subseteq m_1..n_1$. We distinguish two cases: either $n_1 = \infty$ or $n_1 \in \mathbb{N}^+$.

If $n_1 = \infty$, then it must be $m_2 < m_1$. Since $M \parallel A \sqsubseteq_{m_2..n_2} M$, by Definition 10 there is a trace t, with $\#\mathsf{tick}(t) = m_2{-}1$, such that $M \parallel A \xrightarrow{t}$ and $M \stackrel{\hat{t}}{\not\Longrightarrow}$. By Lemma 4, this entails $M \parallel \mathit{Top}(C) \stackrel{\hat{t}}{\Longrightarrow}$. Since $M \stackrel{\hat{t}}{\not\Longrightarrow}$ and $\#\mathsf{tick}(t) = m_2{-}1 < m_2 < m_1$, this contradicts $M \parallel \mathit{Top}(C) \sqsubseteq_{m_1..n_1} M$.

If $n_1 \in \mathbb{N}^+$, then $m_2 < m_1$ and/or $n_1 < n_2$, and we reason as in the previous case. □

Proof of Theorem 4. We consider the two parts of the statement separately.

Definitive impact. By an application of Lemma 4 we have that $M \parallel A \xrightarrow{t}$ entails $M \parallel \mathit{Top}(C) \stackrel{\hat{t}}{\Longrightarrow}$. This implies $M \parallel A \sqsubseteq M \parallel \mathit{Top}(C)$. Thus, if $M \parallel \mathit{Top}(C) \sqsubseteq M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\xi]$, for $\xi \in \mathbb{R}^{\hat{\mathcal{X}}}$, $\xi > 0$, then, by transitivity of $\sqsubseteq$, it follows that $M \parallel A \sqsubseteq M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\xi]$.

Pointwise impact. The proof proceeds by contradiction. Suppose $\xi' > \xi$. Since $\mathit{Top}(C)$ has a pointwise impact ξ at time m, it follows that ξ is given by:

$$\inf\big\{\xi'' : \xi'' \in \mathbb{R}^{\hat{\mathcal{X}}} \;\wedge\; M \parallel \mathit{Top}(C) \sqsubseteq_{m..n} M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\xi''], n \in \mathbb{N}^+ \cup \infty\big\}.$$

Similarly, since A has a pointwise impact ξ' at time m', it follows that ξ' is given by

$$\inf\big\{\xi'' : \xi'' \in \mathbb{R}^{\hat{\mathcal{X}}} \;\wedge\; M \parallel A \sqsubseteq_{m'..n} M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\xi''], n \in \mathbb{N}^+ \cup \infty\big\}.$$

Now, if $m = m'$, then $\xi \geq \xi'$ because $M \parallel A \xrightarrow{t}$ entails $M \parallel \mathit{Top}(C) \stackrel{\hat{t}}{\Longrightarrow}$ due to an application of Lemma 4. This is contradiction with the fact that $\xi < \xi'$. Thus, it must be $m' < m$. Now, since both ξ and ξ' are the infimum functions and since $\xi' > \xi$, there are $\overline{\xi}$ and $\overline{\xi'}$, with $\xi \leq \overline{\xi} \leq \xi' \leq \overline{\xi'}$ such that: (i) $M \parallel \mathit{Top}(C) \sqsubseteq_{m..n} M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\overline{\xi}]$, for some n; (ii) $M \parallel A \sqsubseteq_{m'..n'} M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\overline{\xi'}]$, for some n'.

From $M \parallel A \sqsubseteq_{m'..n'} M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}+\overline{\xi'}]$ it follows that there exists a trace t with $\#\mathsf{tick}(t) = m' - 1$ such that $M \parallel A \xrightarrow{t}$ and $M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}+\overline{\xi'}] \stackrel{\hat{t}}{\not\Longrightarrow}$. Since $\overline{\xi} \leq \overline{\xi'}$, by monotonicity (Proposition 6), we deduce that $M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}+\overline{\xi}] \stackrel{\hat{t}}{\not\Longrightarrow}$. Moreover, by Lemma 4 $M \parallel A \xrightarrow{t}$ entails $M \parallel \mathit{Top}(C) \stackrel{\hat{t}}{\Longrightarrow}$.

Summarising, there exists a trace t' with $\#\mathsf{tick}(t') = m' - 1$ such that $M \parallel \mathit{Top}(C) \xrightarrow{t'}$ and $M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}+\overline{\xi}] \stackrel{\hat{t'}}{\not\Longrightarrow}$. However, this, together with $m' < m$, is in contradiction with the fact (i) above saying that $M \parallel \mathit{Top}(C) \sqsubseteq_{m..n} M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}+\overline{\xi}]$, for some n. As a consequence it must be $\xi' \leq \xi$ and $m' \leq m$. This concludes the proof. □